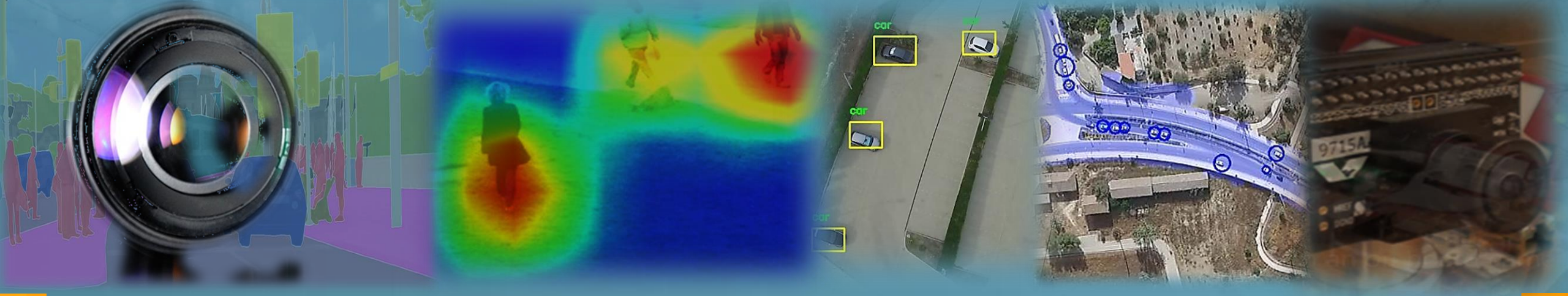


Lecture 5: Image Feature Detectors and Descriptors

Dr Christos Kyrkou

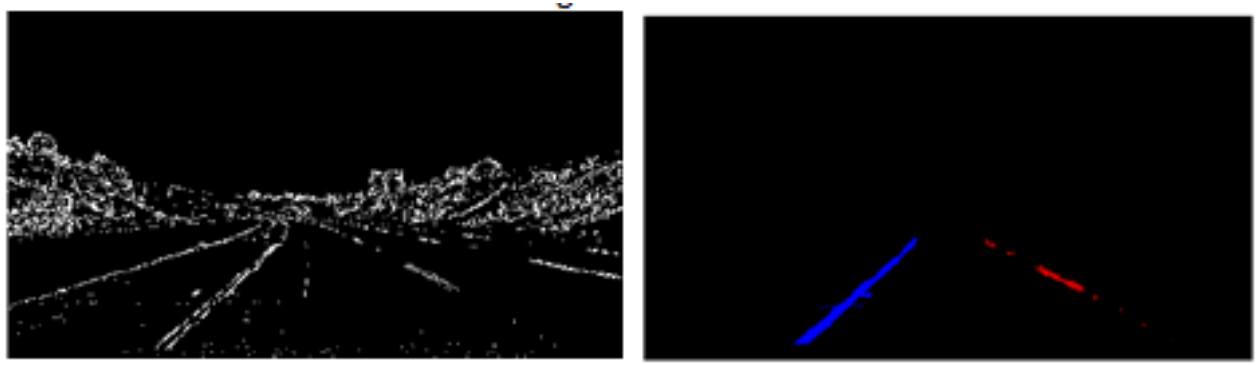
KIOS Research and Innovation Center of Excellence,

Electrical and Computer Engineering Department, University of Cyprus

kyrkou.christos@ucy.ac.cy

Previously

- Edge detection, Sobel, Canny
- RANSAC and Hough Transform

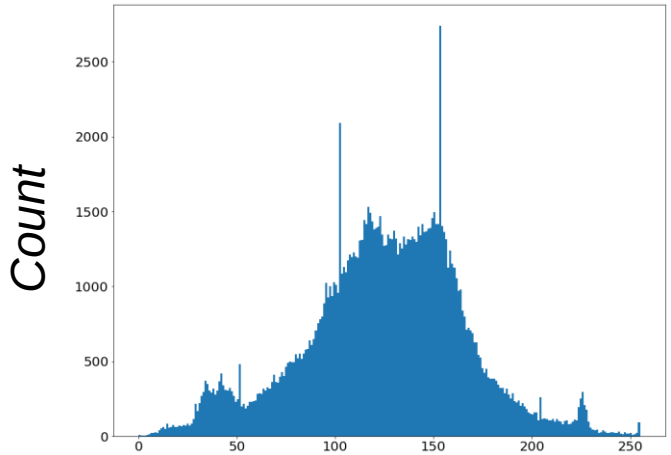


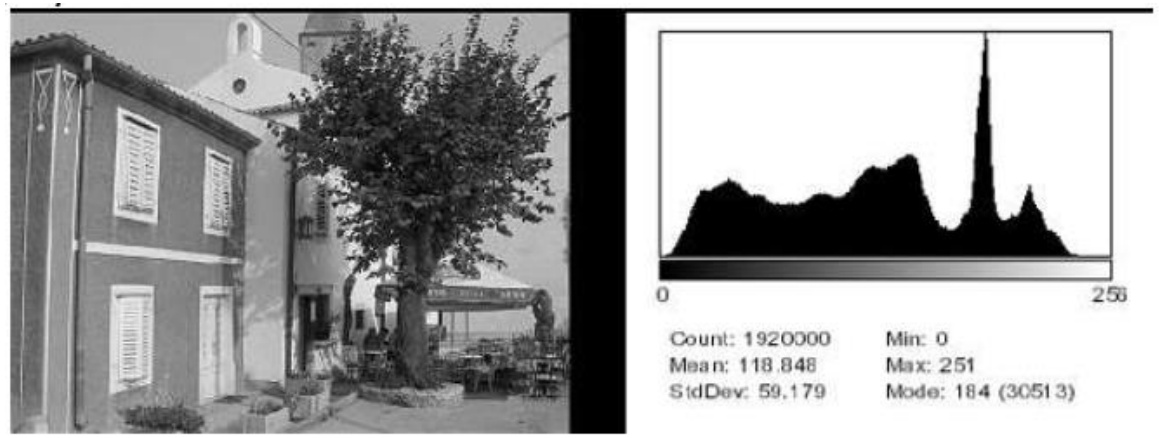
Outline

- Histograms
- Harris Corner Detector
- Scale invariant region selection
  - Automatic scale selection
  - Difference-of-Gaussian (DoG) detector
- SIFT: An image region descriptor
- HOG: Another image descriptor
- Local Binary Patterns (LBP): Descriptor for textures

Histograms

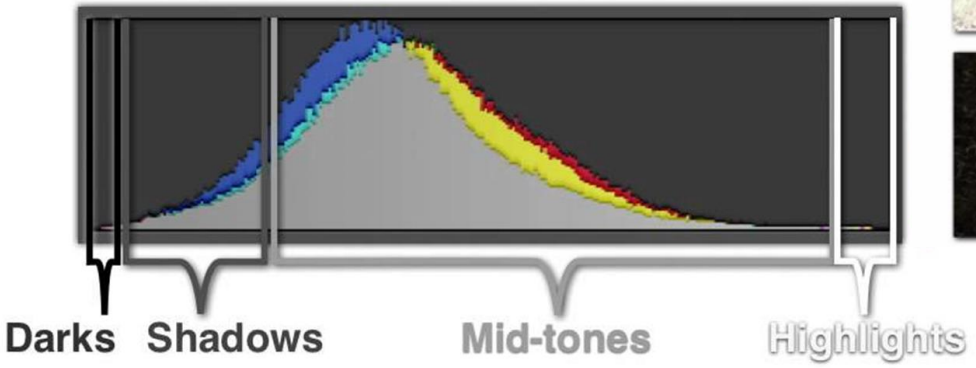
- Image has 256 distinct gray-levels (8bits)
- Histogram shows frequency (how many times) each gray-level occurs
  - Captures the distribution of gray levels in the image.

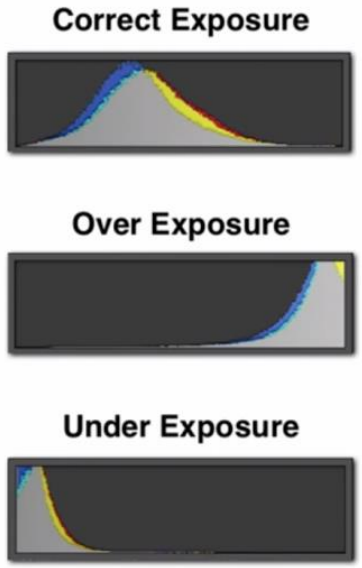


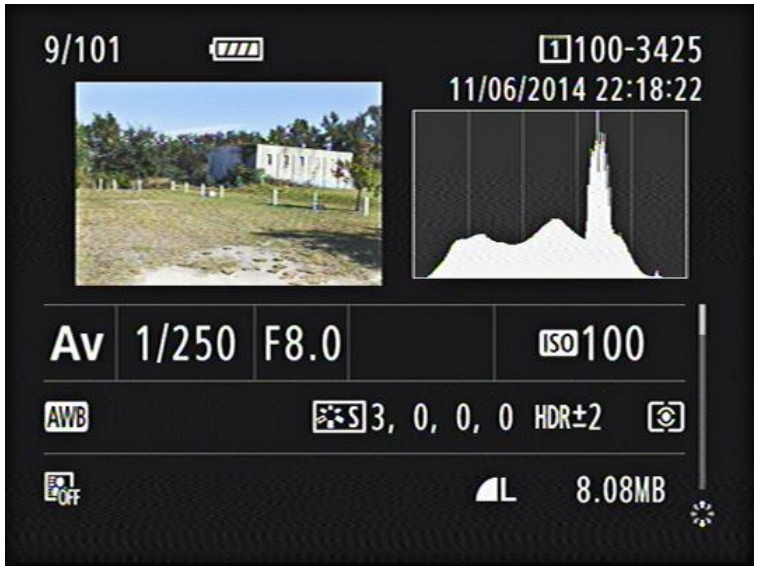


Histograms

- Many cameras display real time histograms of scene
  - Helps avoid taking over-exposed pictures
  - Also easier to detect types of processing previously applied to image

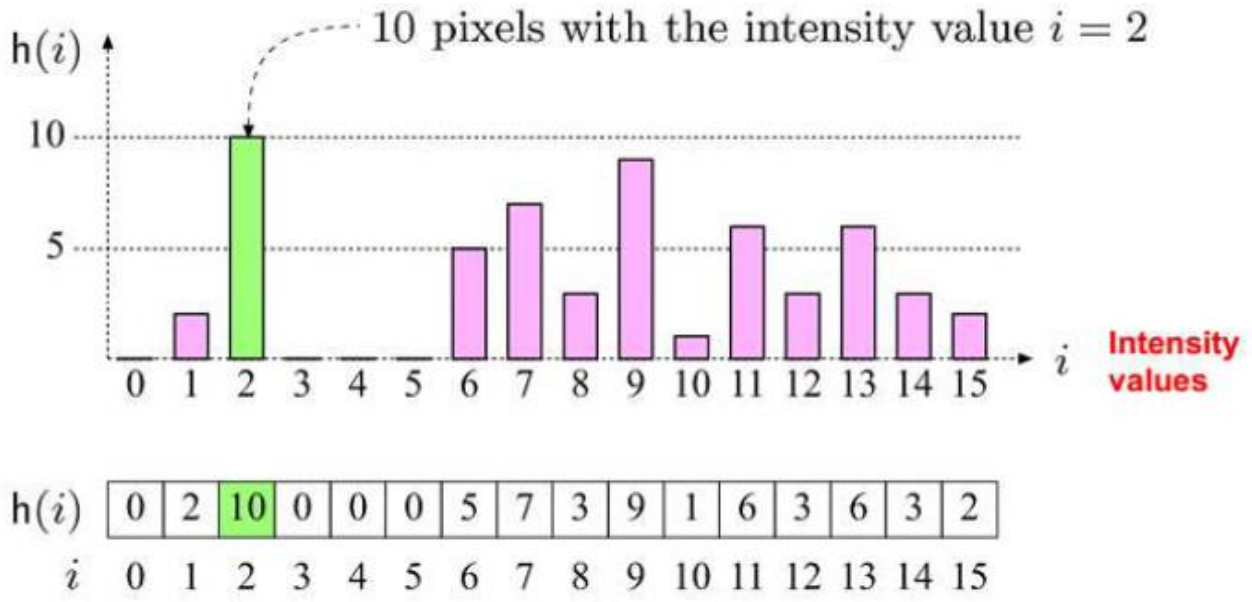






Histogram Example

- E.g.
  - 16 intensity levels
  - 10 pixels have intensity value = 2
  - These 10 pixels can be in very different parts of an image



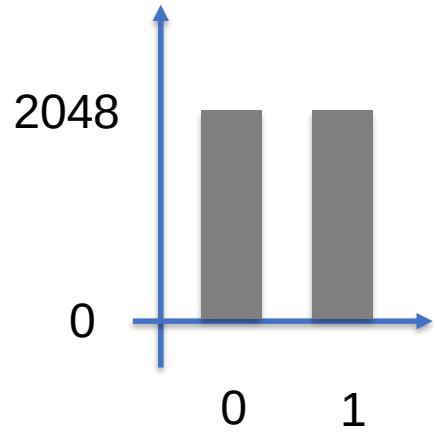
How to construct a Histogram

- A histogram for a grayscale image $I(x, y)$ with intensity values in range $[0, L-1]$ would contain exactly L entries
  - E.g. 8-bit grayscale image, L = 2^8 = 256
- Each histogram entry is defined as:
  - $h(i)$ = number of pixels with intensity i for all $0 \leq i < L$
  - E.g: $h(255)$ = number of pixels with intensity == 255

```
def histogram(im):
    h = np.zeros(255)
    for row in im.shape[0]:
        for col in im.shape[1]:
            val = im[row, col]
            h[val] += 1
```

Histogram Example

- Draw the histogram of following images. Consider each image of size 64x64 (2-levels).



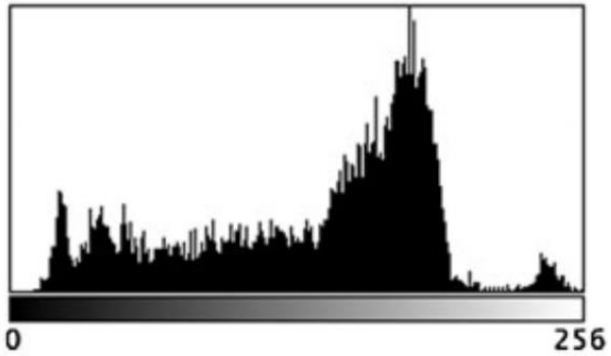
- **Different images can have same histogram**

Histogram Example

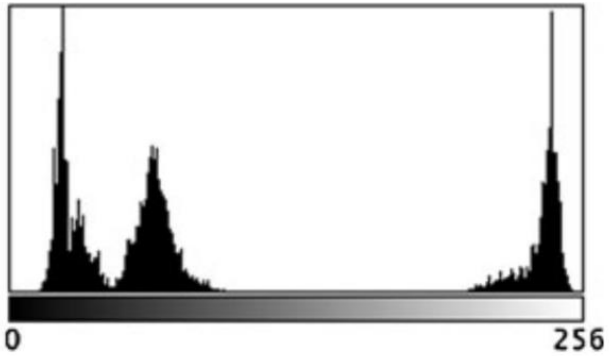
- Half of pixels are gray, half are white
  - Same histogram = same statistics
  - Spatial distribution of intensities could be different

- Can we reconstruct image from histogram?
  - No, spatial information is lost!
  - Histograms: only statistical information
    - No indication of location of pixels

Histogram as Features



Count: 10192
Mean: 133.711
StdDev: 55.391
Min: 9
Max: 255
Mode: 178 (180)

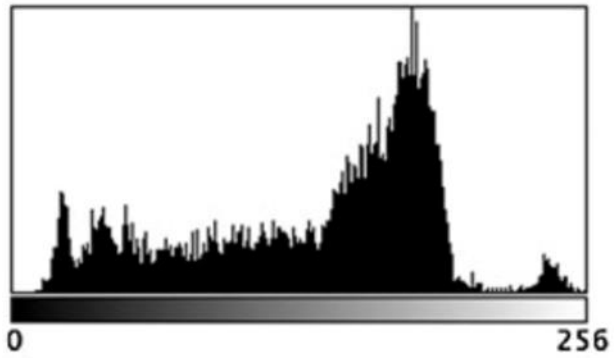


Count: 10192
Mean: 104.637
StdDev: 89.862
Min: 11
Max: 254
Mode: 23 (440)

Slide credit: Dr. Mubarak Shah

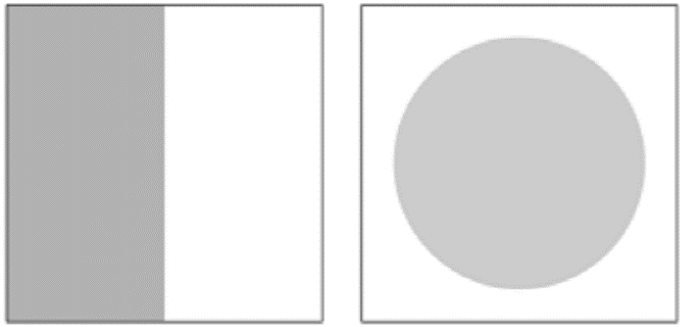
Histogram Matching

- Histograms provide a neat way of describing an image region



- But different images can have the same histogram

Over-reliance on Pixel Values



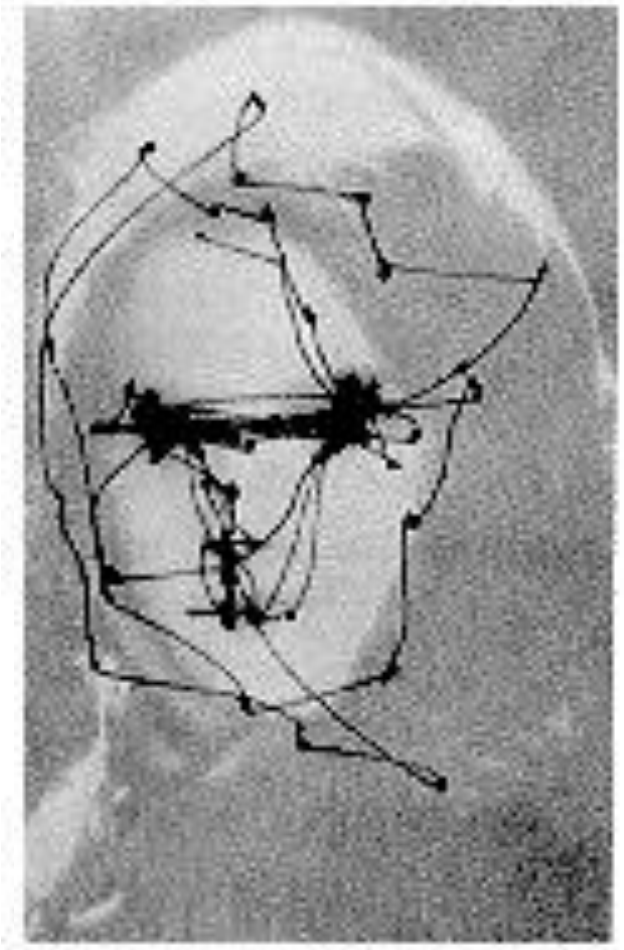
Problems with pixel representation

- Not invariant to small changes
  - Translation
  - Illumination
  - etc.

- Some parts of an image are more important than others

- What do we want to represent?
  - Shape-boundary
  - Color and texture information

Human eye movements

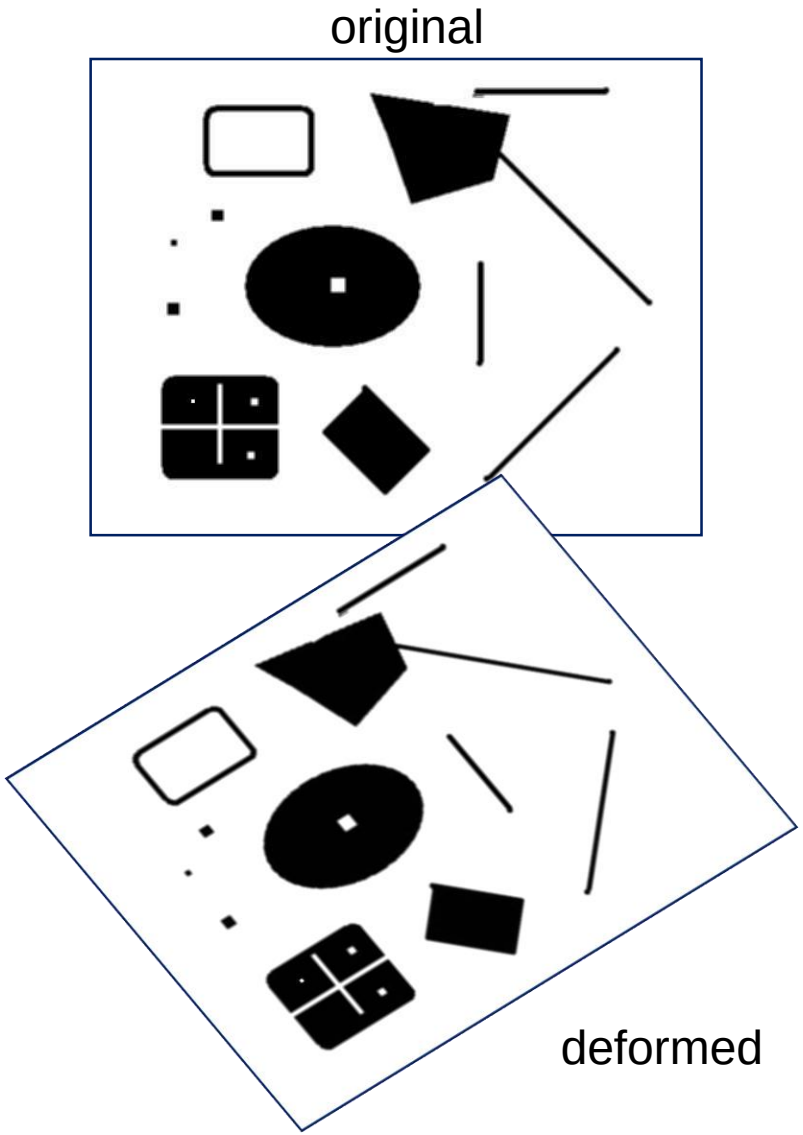


Yarbus eye tracking

What catches your interest?

Choosing interesting points

- Suppose you have to click on some point, go away and come back after I deform the image, and click on the same points again.
  - Which points would you choose?



Choosing interesting points

- Where would you tell your friend to meet you?

→ Corner detection

D. Hoiem

Choosing interesting points

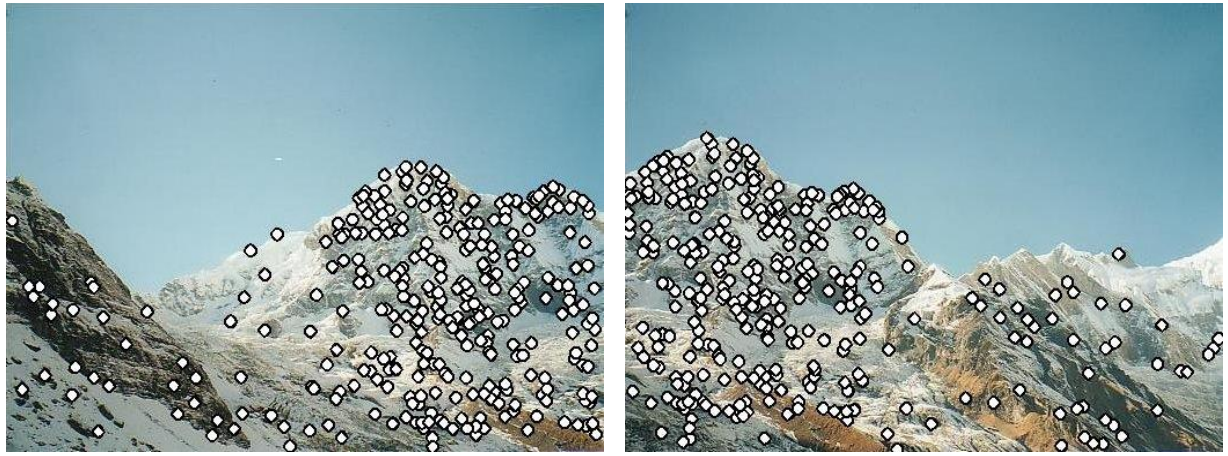
- Where would you tell your friend to meet you?

→ Blob detection

D. Hoiem

Interest(ing) points

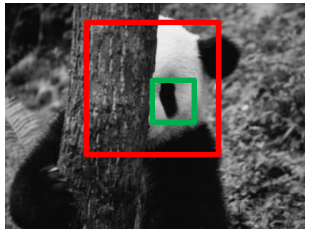
- Note: “interest points” = “keypoints”, also sometimes called “features”
- Usually they only cover a small part of the image
- There will be many local features detected in an image
- Local features usually exploit image gradients, ignore color

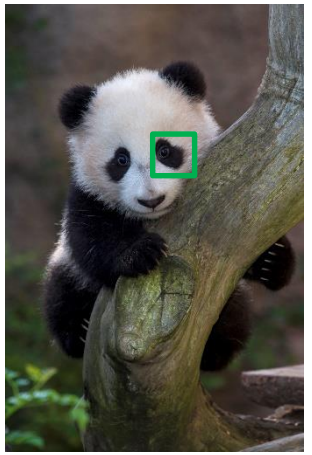


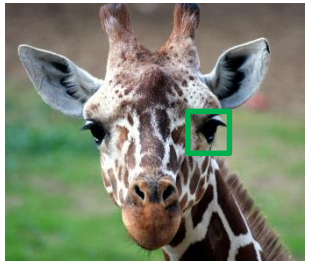
Adapted from D. Hoiem

Local features: desired properties

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



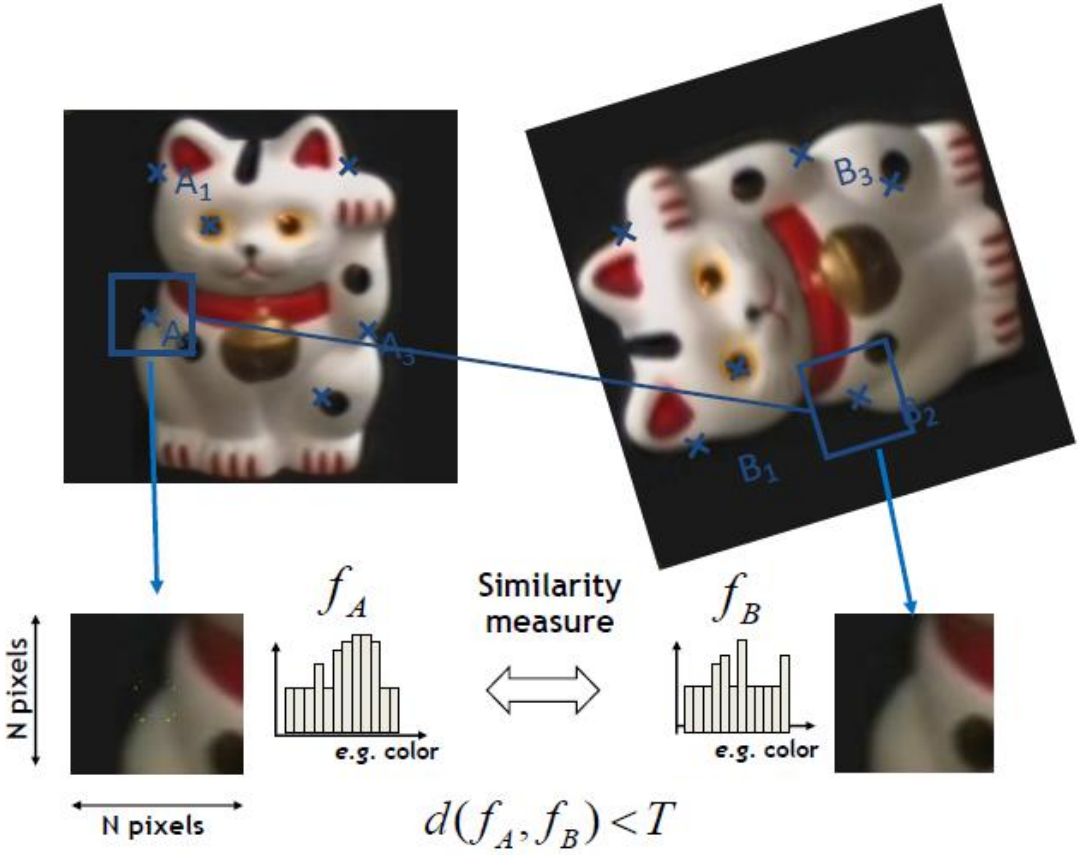




Adapted from K. Grauman and D. Hoiem

General Approach

- Find a set of distinctive key-points
- Define a region around each **keypoint**
- Extract and normalize the region content
- Compute a local **descriptor** from the normalized region
- Match local **descriptors**



Applications

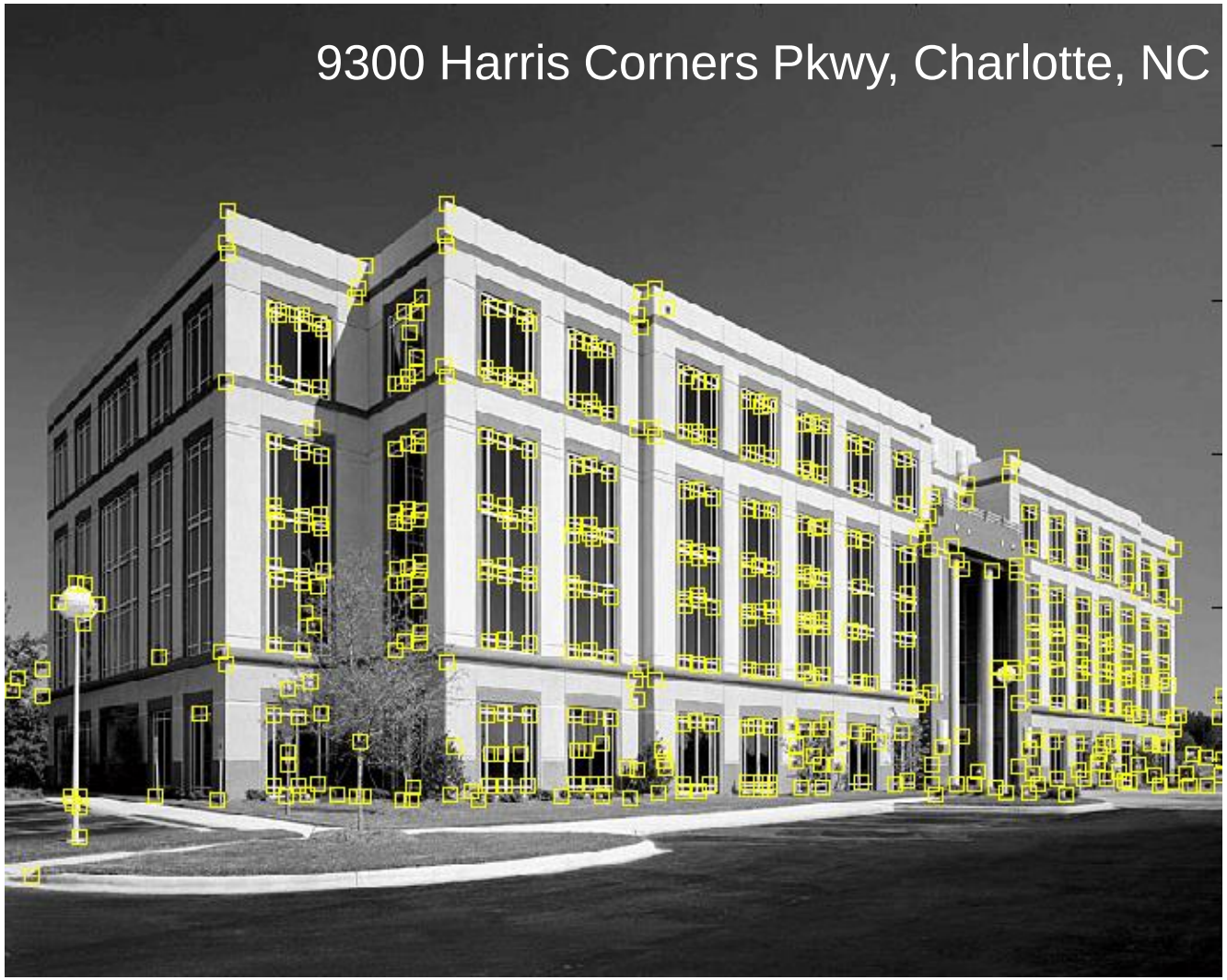
- **Image alignment:** warping one image (or sometimes both images) so that the features in the two images line up perfectly.
- **3D reconstruction:** how to find correspondences across different views for computing disparity.
- **Motion tracking:** which points are good to track? Automate object tracking in a video.
- **Robot navigation:** Analyze motion to find out where it is in the world. Create a map.
- **Recognition:** which patches are likely to tell us something about the object category?
- **Database indexing and retrieval:** which points would allow us to match images between query and database?

HARRIS CORNER DETECTOR

Section 4.1.1Feature Detectors, Richard Szeliski, “Computer Vision: Algorithms and Applications”, Springer

Keypoint extraction: Corners

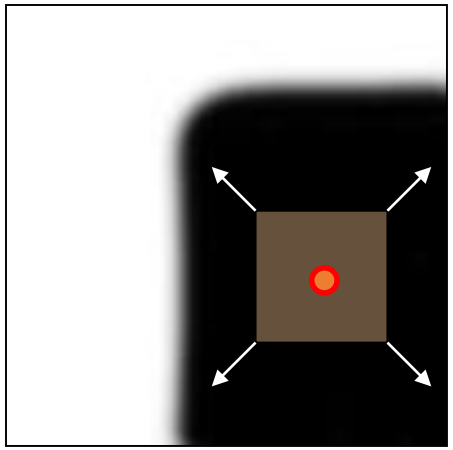
- Harris operator is one of the most simple, efficient and reliable for use in corner detection
- Harris corner detector is an important and fundamental technique for many computer vision applications.
- If we use Harris corner detector in a color image, the first step is to convert it into a grayscale image, which will enhance the processing speed.



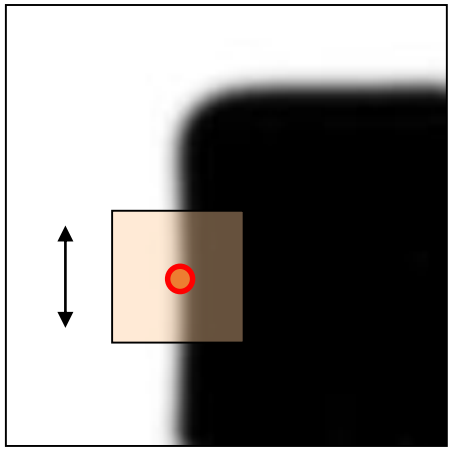
Corners as distinctive interest points

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

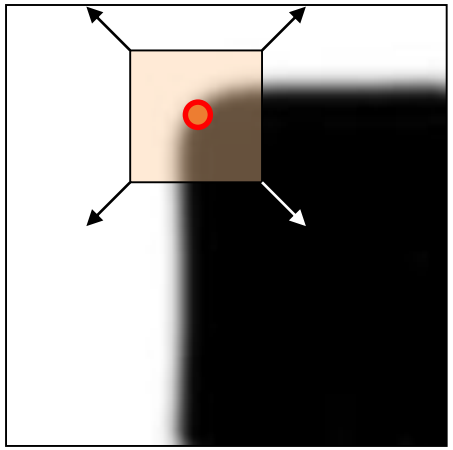
Candidate keypoint



"flat" region:
no change in all directions



"edge":
no change along the edge direction

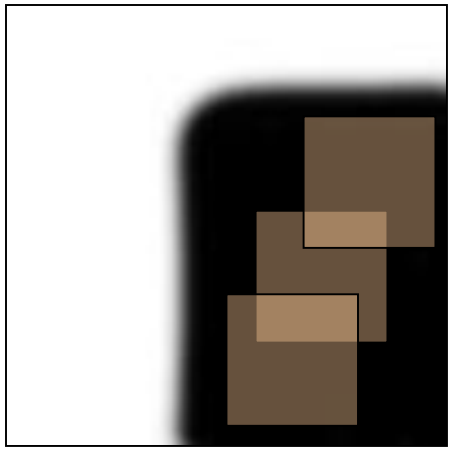


"corner":
significant change in all directions

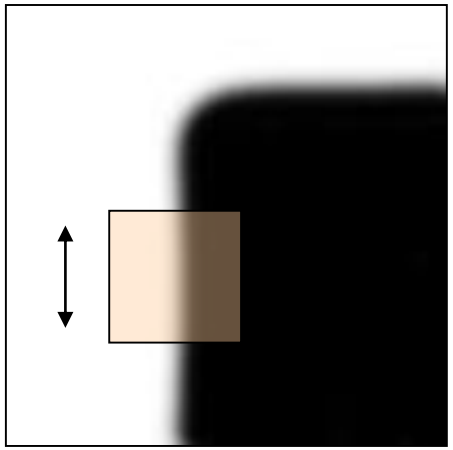
Adapted from A. Efros, D. Frolova, D. Simakov

Corners as distinctive interest points

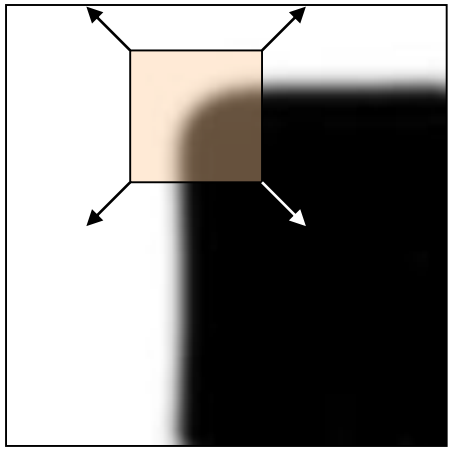
- We should easily recognize the keypoint by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



"flat" region:
no change in
all directions



"edge":
no change along
the edge direction

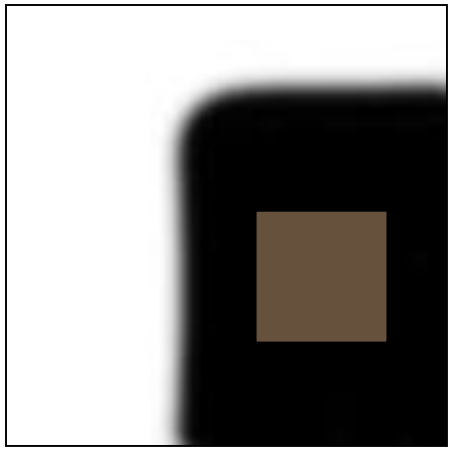


"corner":
significant change
in all directions

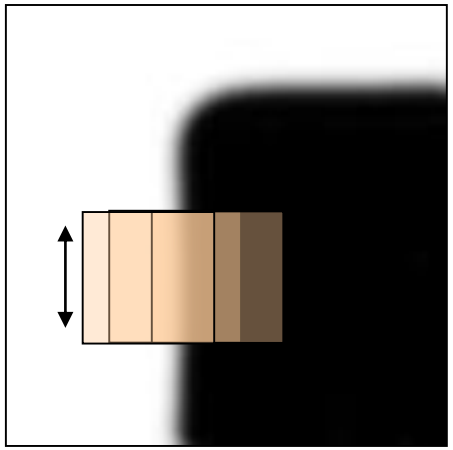
Adapted from A. Efros, D. Frolova, D. Simakov

Corners as distinctive interest points

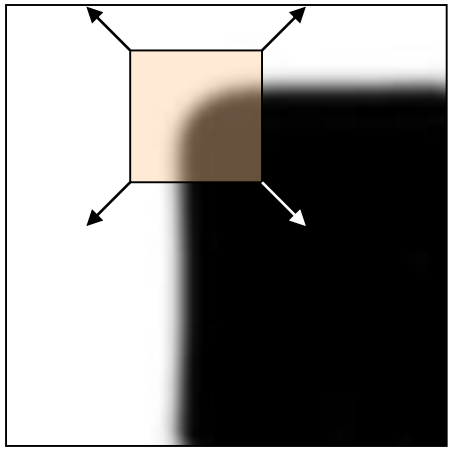
- We should easily recognize the keypoint by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



"flat" region:
no change in
all directions



"edge":
no change along
the edge direction

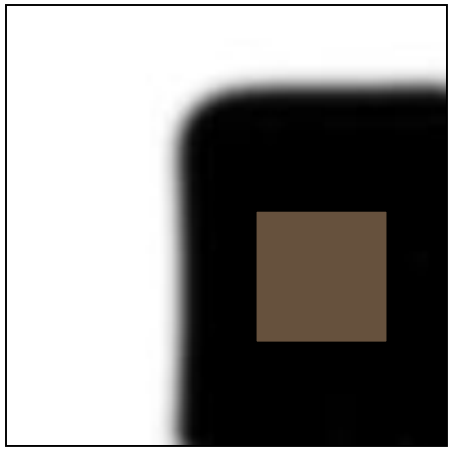


"corner":
significant change
in all directions

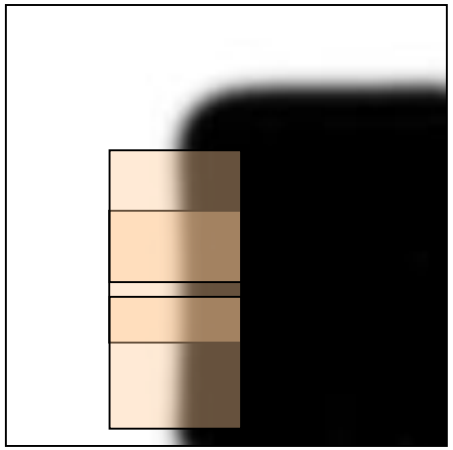
Adapted from A. Efros, D. Frolova, D. Simakov

Corners as distinctive interest points

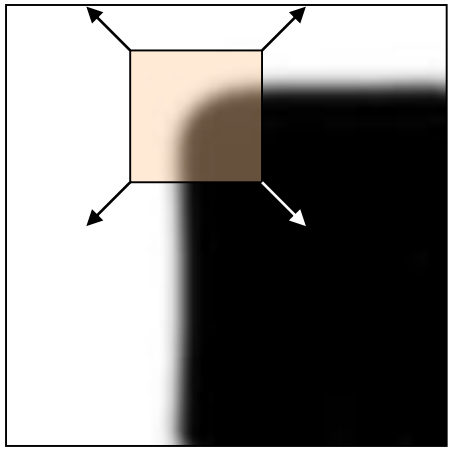
- We should easily recognize the keypoint by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



"flat" region:
no change in
all directions



"edge":
no change along
the edge direction

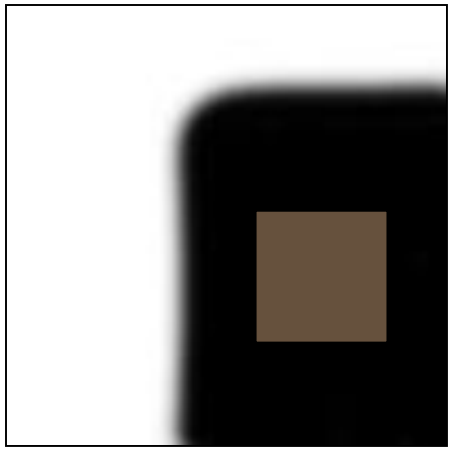


"corner":
significant change
in all directions

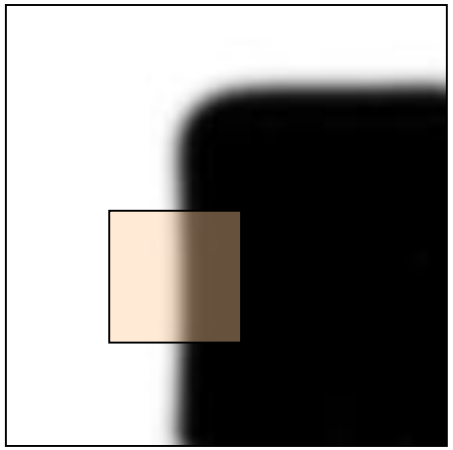
Adapted from A. Efros, D. Frolova, D. Simakov

Corners as distinctive interest points

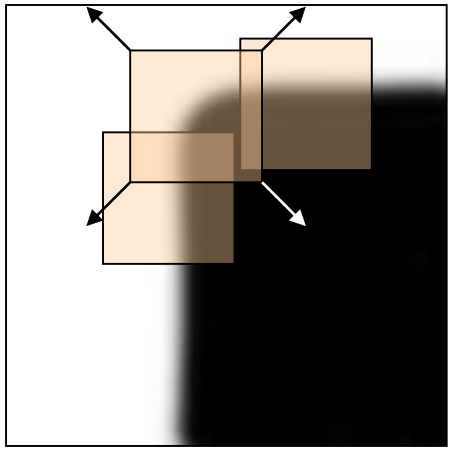
- We should easily recognize the keypoint by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



"flat" region:
no change in
all directions



"edge":
no change along
the edge direction



"corner":
significant change
in all directions

Adapted from A. Efros, D. Frolova, D. Simakov

What points would you choose?

K. Grauman

Harris Detector: Mathematics

- Window-averaged squared change of intensity induced by shifting the patch for a fixed candidate keypoint by [u,v]:

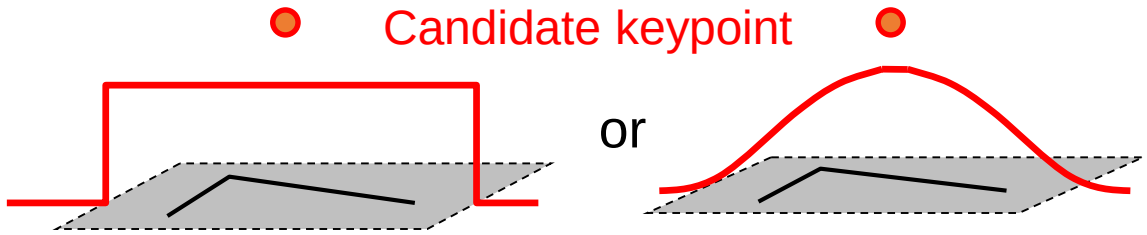
$$E(u,v) = \sum_{x,y} \left[I(x+u, y+v) - I(x,y) \right]^2$$

Window function over neighbors *(x, y)*

Shifted intensity

Intensity

Window function $w(x,y)$ =



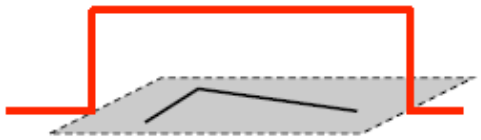
1 in window, 0 outside

Gaussian

D. Frolova, D. Simakov

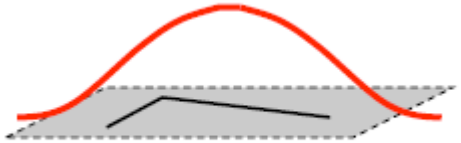
Window Function

- Option 1:
  - Uniform window Sum over square window
  - Problem:
    - Not rotation invariant



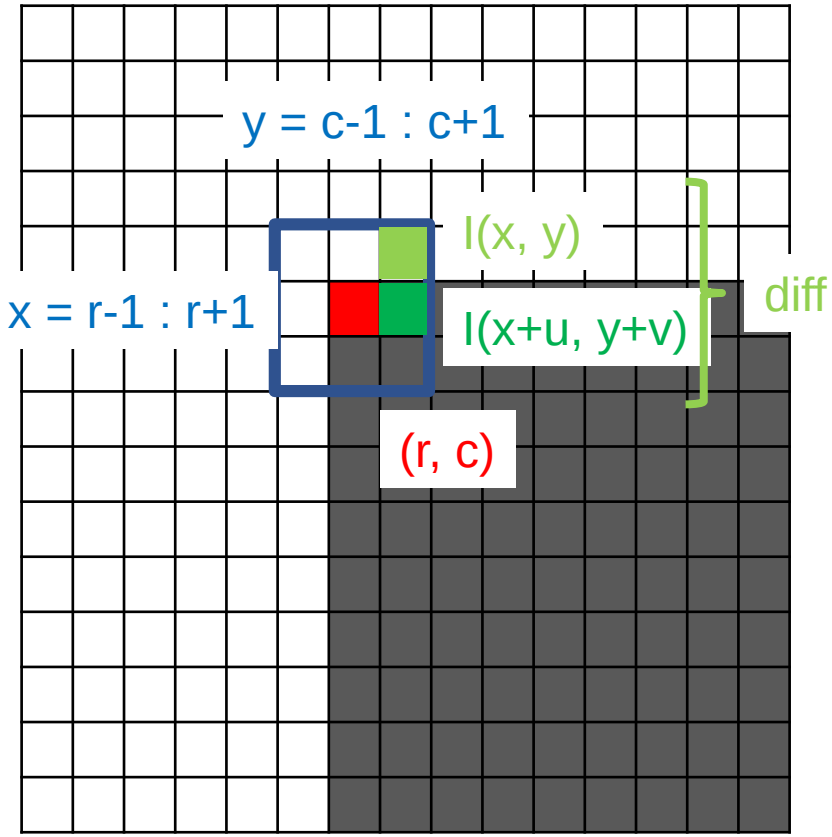
1 in window, 0 outside

- Option 2:
  - Smooth with Gaussian
  - Gaussian already performs weighted sum
  - Result is rotation invariant



Gaussian

Computing the squared change



Here u = 1, v = 0

For every pixel (r, c) as candidate keypoint

Initialize E = zeros(max_offset, max_offset)

For each offset (u, v)

Initialize *sum* to 0

For each neighbor (x, y) of (r, c)

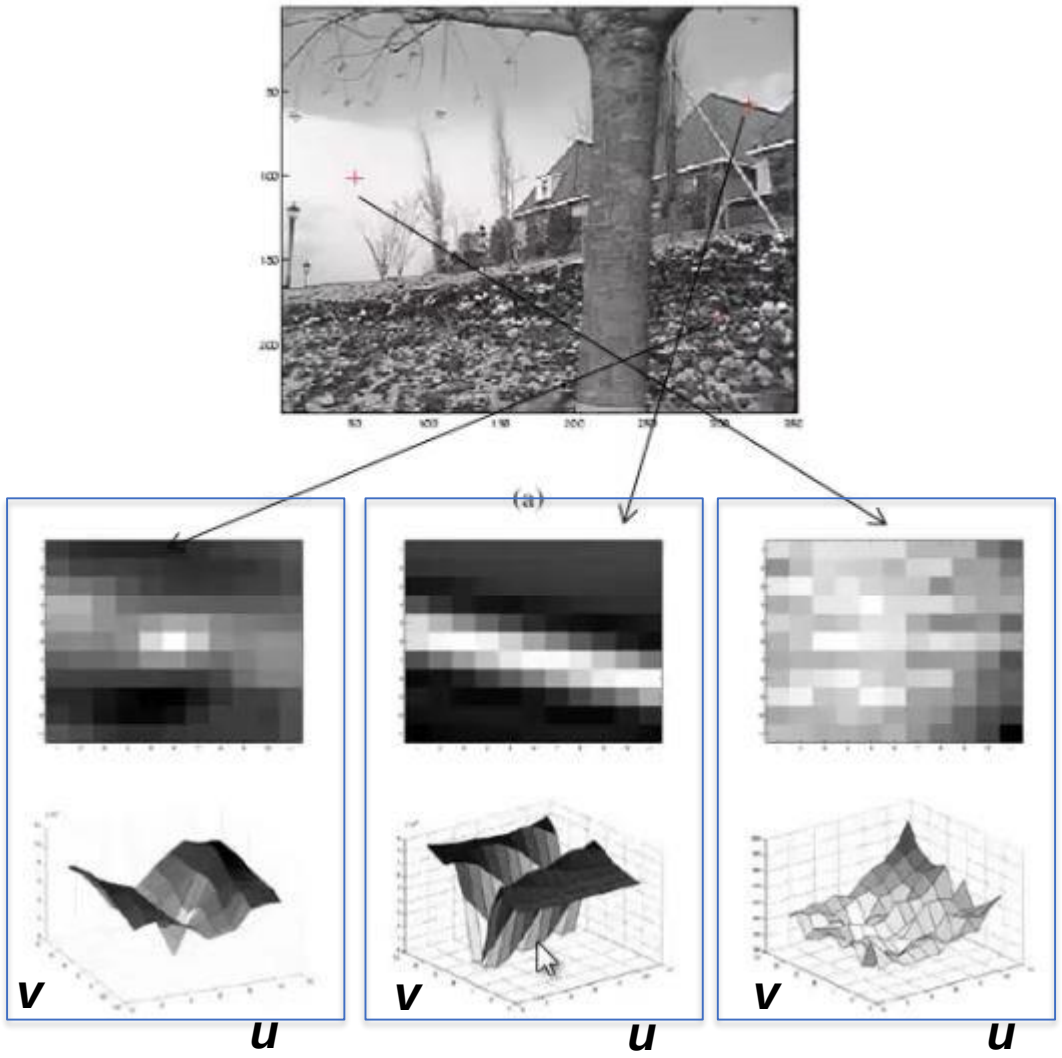
$sum = sum + [I(x, y) - I(x+u, y+v)]^2$

E(u, v) = *sum*

Plot E(u, v)

Visualization

- Plot of the $E(u, v)$ surface where we displace the image patch
- Clear local minima
- Same for edges
  - Many local minima
- Flat regions are ambiguous
  - No clear pattern



Harris Corner Detector: Derivation

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

$$I(x+u, y+v) \approx I(x,y) + uI_x(x,y) + vI_y(x,y)$$

First order Taylor series approximation

$$\begin{aligned}
& [I(x+u, y+v) - I(x,y)]^2 \quad \leftarrow \text{Recall the square term of } E(u,v) \\
\approx\ & [I(x,y) + uI_x(x,y) + vI_y(x,y) - I(x,y)]^2 \\
=\ & [uI_x(x,y) + vI_y(x,y)]^2 = [uI_x + vI_y]^2 \\
=\ & u^2 I_x^2 - v^2 I_y^2 + 2uvI_xI_y \quad \text{Removing (x,y) for readability} \\
=\ & \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}
\end{aligned}$$

$f(x)$ Can be represented at point *a* in terms of its derivatives:

First order term only!

$$f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f^{(3)}(a)}{3!}(x-a)^3 + \cdots.$$

Harris Detector: Mathematics

- For small shifts [u,v], the measure of change can be approximated by:
  - First-order Taylor approximation

$$E(u,v) \cong \begin{bmatrix} u, v \end{bmatrix} \; M \; \begin{bmatrix} u \\ v \end{bmatrix}$$

Equation of an elipse

- where M is a 2×2 matrix computed from image derivatives:
  - Second Moment Matrix

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_h^2 & I_h I_v \\ I_h I_v & I_v^2 \end{bmatrix}$$

Gradient with respect to x (Horizontal)

Product of Gradient Components

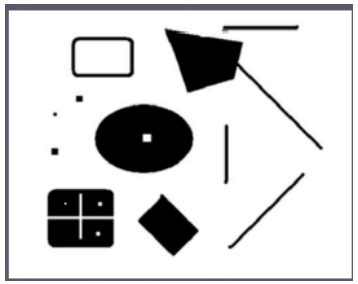
Gradient with respect to y (Vertical)

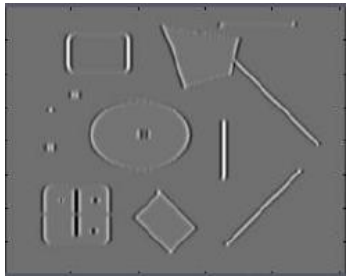
Adapted from A. Efros, D. Frolova, D. Simakov

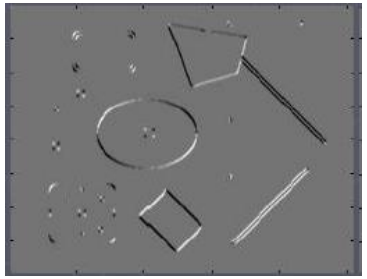
Harris Detector: Mathematics

- How else can I write this, if w(x, y) = 1 inside, 0 outside?

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_h^2 & I_h I_v \\ I_h I_v & I_v^2 \end{bmatrix}$$





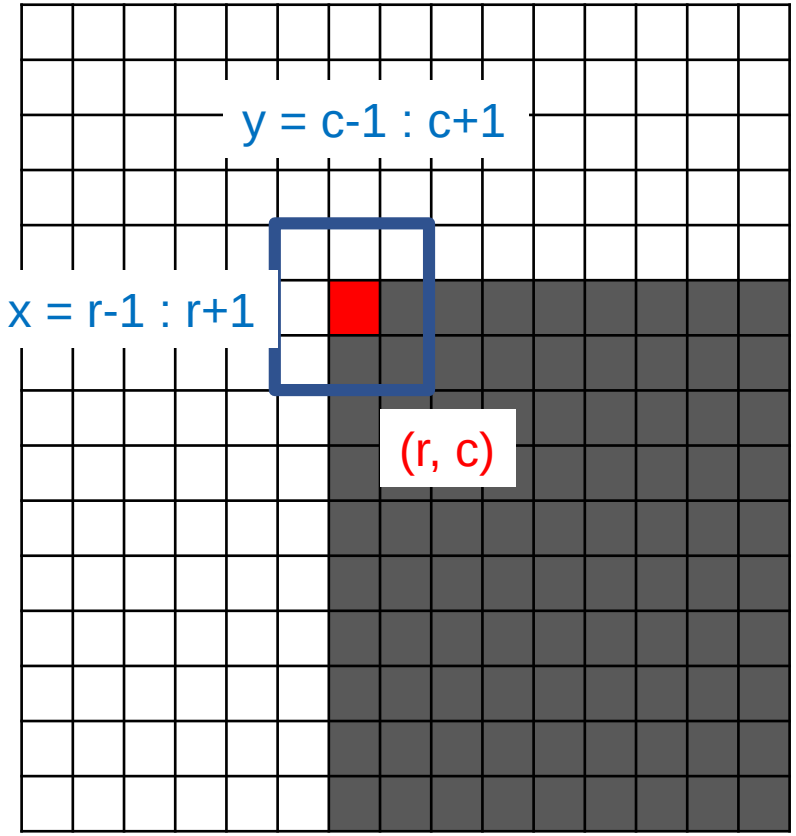


Notation:

$$I_h \Leftrightarrow \frac{\partial I}{\partial x} \qquad I_v \Leftrightarrow \frac{\partial I}{\partial y} \qquad I_h I_v \Leftrightarrow \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}$$

Adapted from K. Grauman

Computing with Second Moment Matrix



Let I_h (of size width x height of the image) be the image derivative in the horizontal direction,
I_v be derivative in the vertical direction.
(Both require one correlation op to compute.)

For every pixel (r, c) as candidate keypoint

Initialize M = zeros(2, 2)

For x = r-1 : r+1

For y = c-1 : c+1

$M(1, 1) = M(1, 1) + I_h(x, y)^2$

$M(1, 2) = ?\ \ M(1, 2) + I_h(x, y) \times I_v(x, y)$

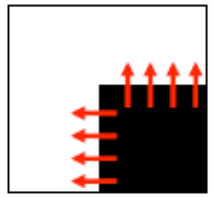
$M(2, 1) = ?\ \ M(1, 2)$

$M(2, 2) = ?\ \ M(2, 2) + I_v(x, y)^2$

Interpreting the second moment matrix

- First, let's consider an axis-aligned corner:

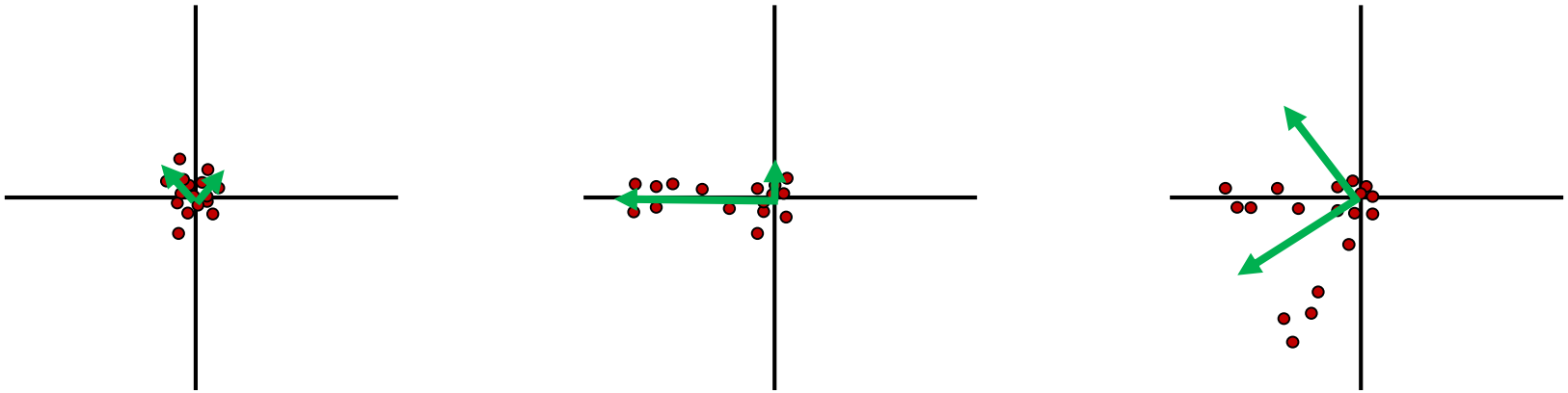
$$M = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$



- This means:
  - Dominant gradient directions align with x or y axis
  - If either λ is close to 0, then this is not a corner, so look for locations where both are large.
- What if we have a corner that is not aligned with the image axes?

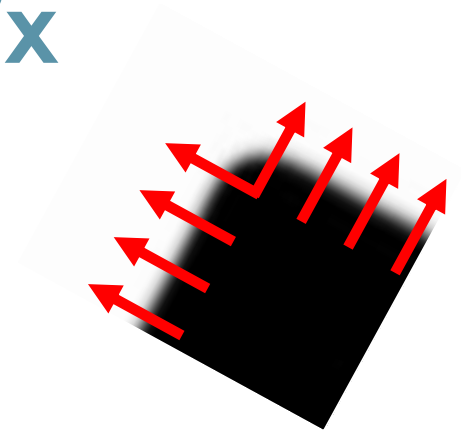
Eigenvectors and Eigenvalues

- Compute eigenvectors and eigenvalues.
- The eigen vector x, of a matrix H is a special vector, with the following property: $Hx = \lambda x$
- The components are the eigenvectors ranked by the eigenvalues λ.
- Highest component is the direction with highest variance orthogonal to the previous components.



Credit: Linda Shapiro

Interpreting the second moment matrix



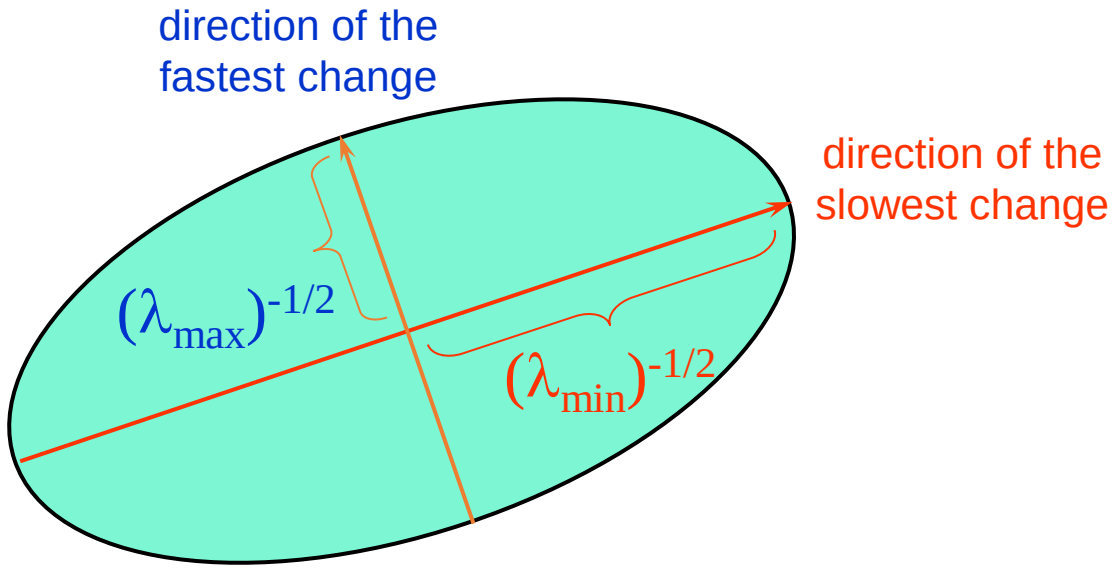
- In the general case, need to *diagonalize* M:
  - Eigenvalue decomposition:

$$M = X \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} X^T$$

$$Mx_i = \lambda_i x_i$$

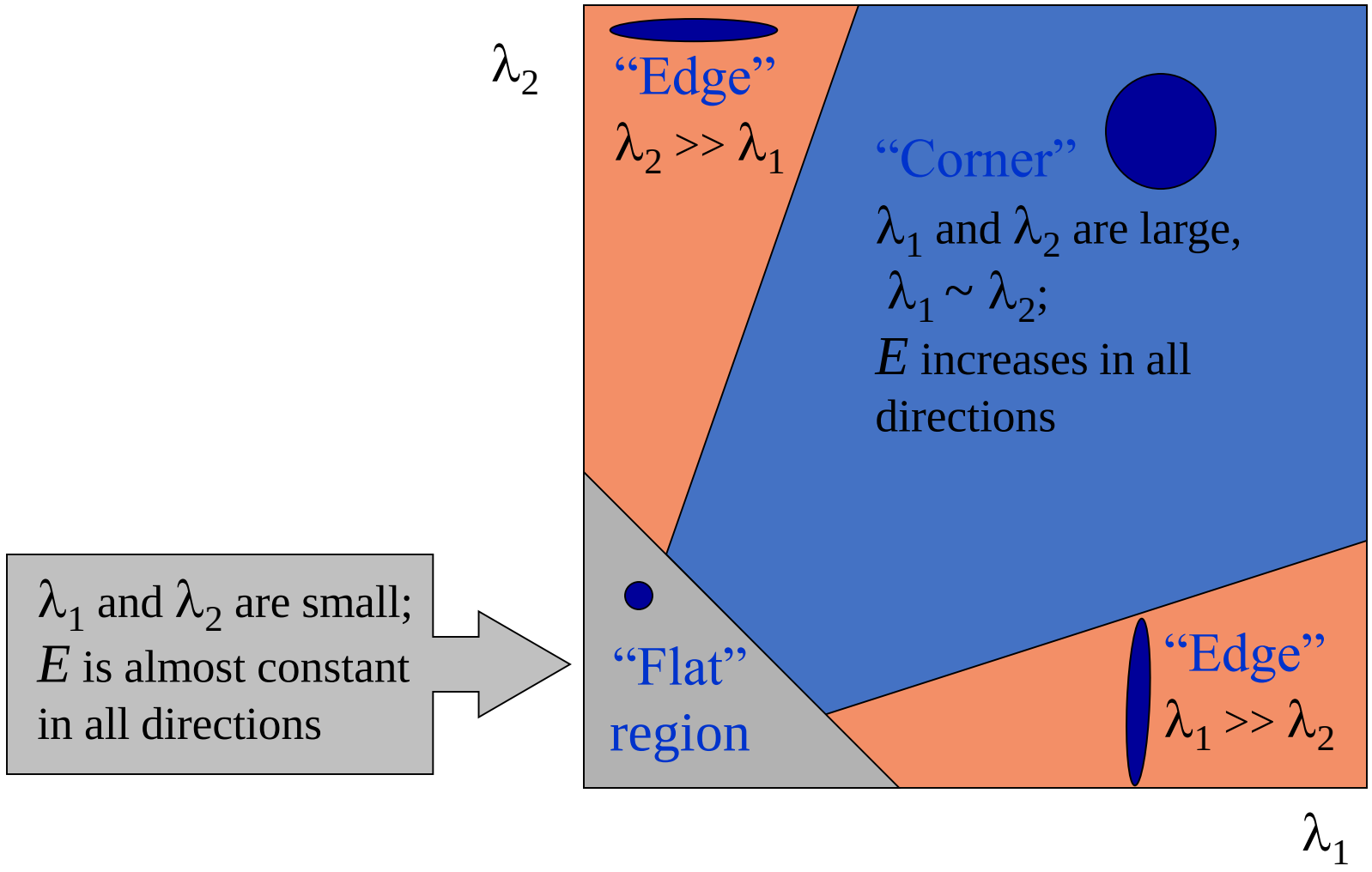
- The axis lengths of the ellipse are determined by the eigenvalues and the orientation is determined by X

The *eigenvalues* of M reveal the amount of intensity change in the two principal orthogonal gradient directions in the window.

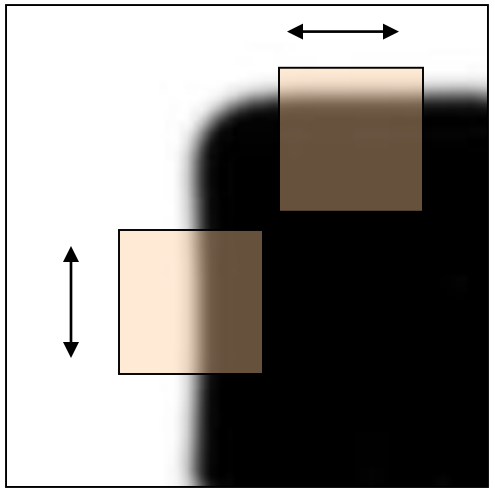


Interpreting the eigenvalues

- Classification of image points using eigenvalues of M:



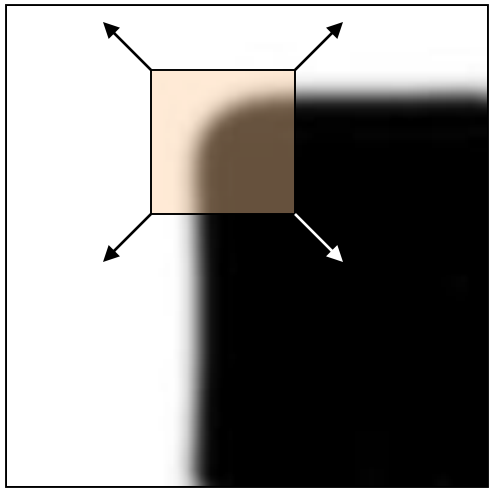
Interpreting the eigenvalues



"edge":

$\lambda_1 >> \lambda_2$

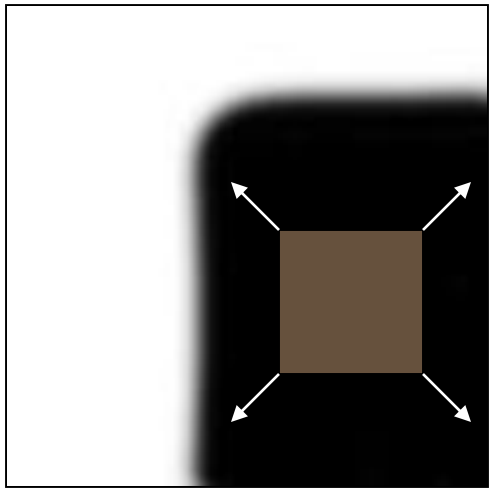
$\lambda_2 >> \lambda_1$



"corner":

λ_1 and λ_2 are large,

$\lambda_1 \sim \lambda_2$



"flat" region:

λ_1 and λ_2 are small

Adapted from A. Efros, D. Frolova, D. Simakov, K. Grauman

Measure of corner response

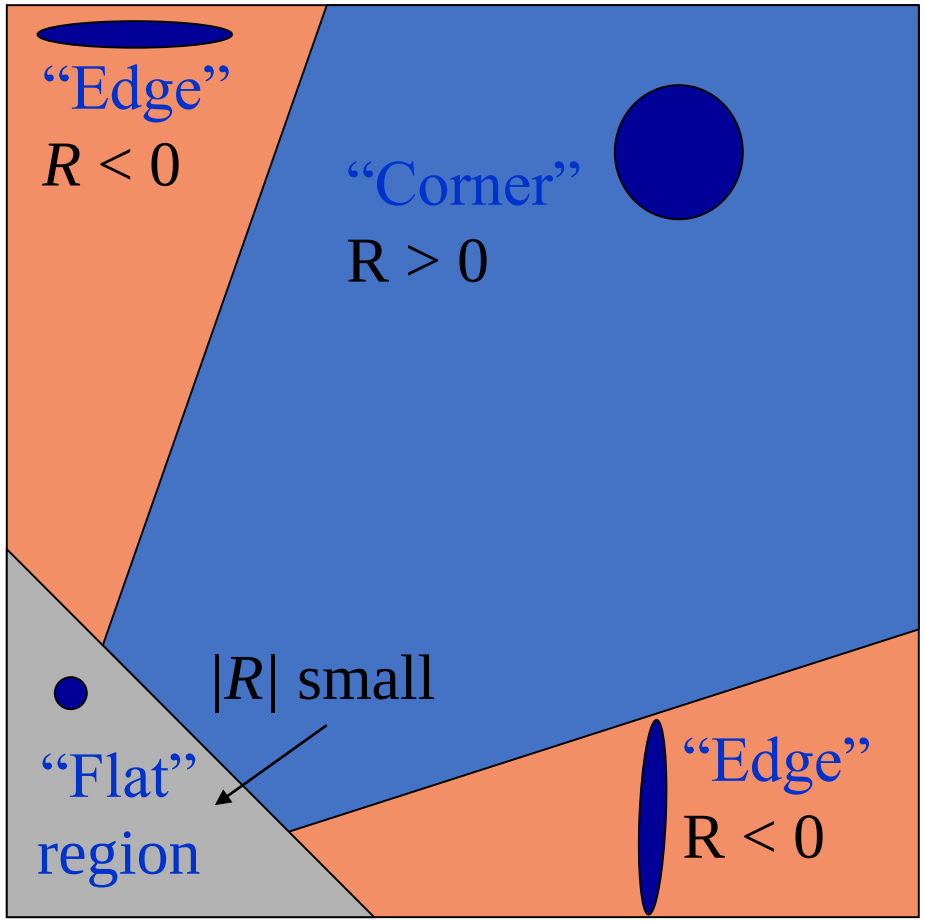
- Fast approximation
  - Avoid computing the eigenvalues

$$R = \det M - k\left(\text{trace}\, M\right)^2$$

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

Because M is symmetric

(k – empirical constant, k = 0.04-0.06)



Reminder:

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

D. Frolova, D. Simakov

Harris Detector: Algorithm

1. Compute image gradients I_h and I_v for all pixels
2. For each pixel
   - Compute:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_h^2 & I_h I_v \\ I_h I_v & I_v^2 \end{bmatrix}$$

   by looping over neighbors x, y

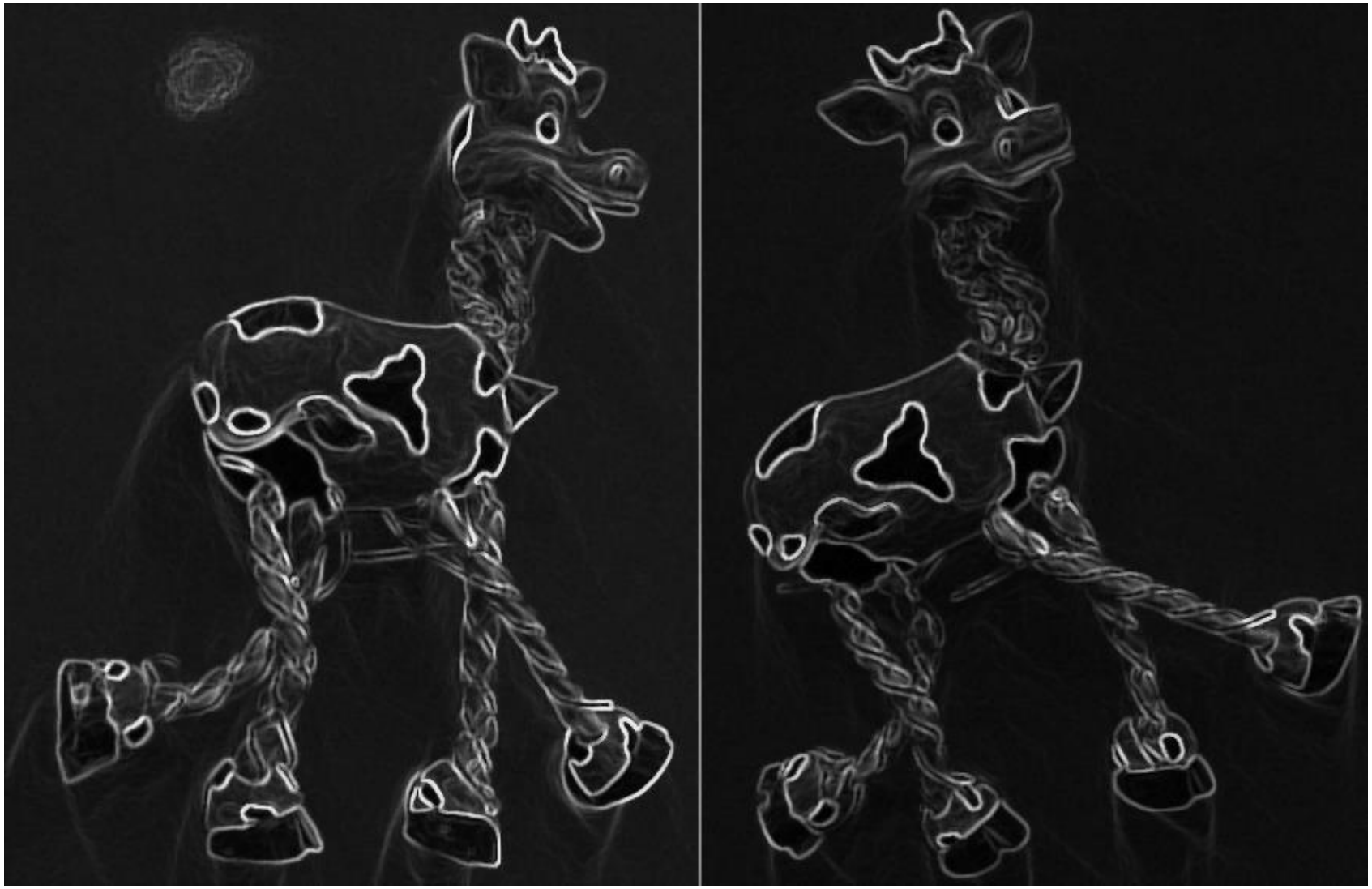
   - compute $R = \det M - k\left(\text{trace } M\right)^2$

3. Find points with large corner response function R
   - (R > threshold)
4. Non-max suppression: Take the points of locally maximum R as the detected feature points (i.e., pixels where R is bigger than for all the 4 or 8 neighbors)

C.Harris and M.Stephens, A Combined Corner and Edge Detector, *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

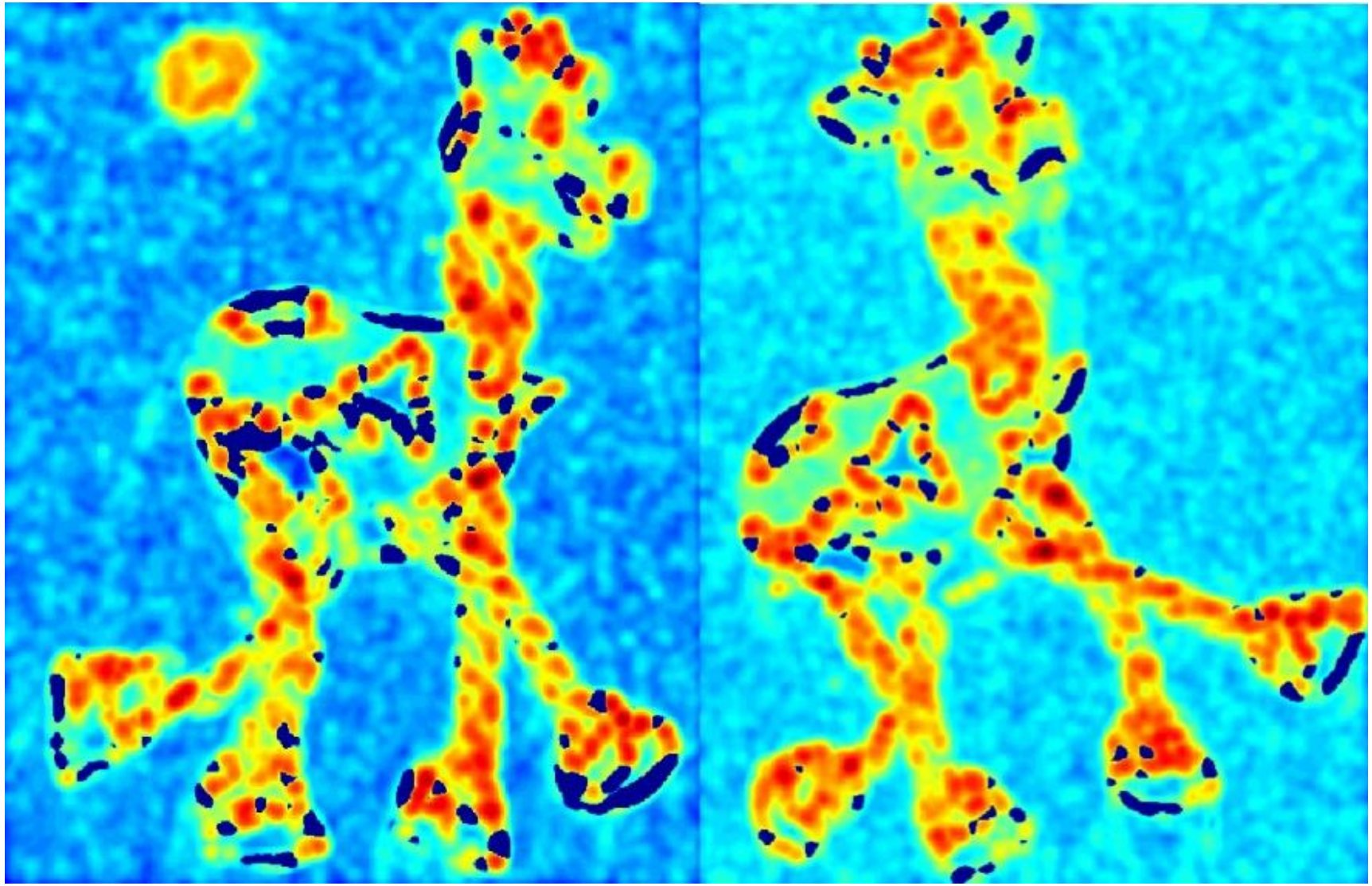
Harris Detector: Steps

- Compute partial derivatives at each pixel



Harris Detector: Steps

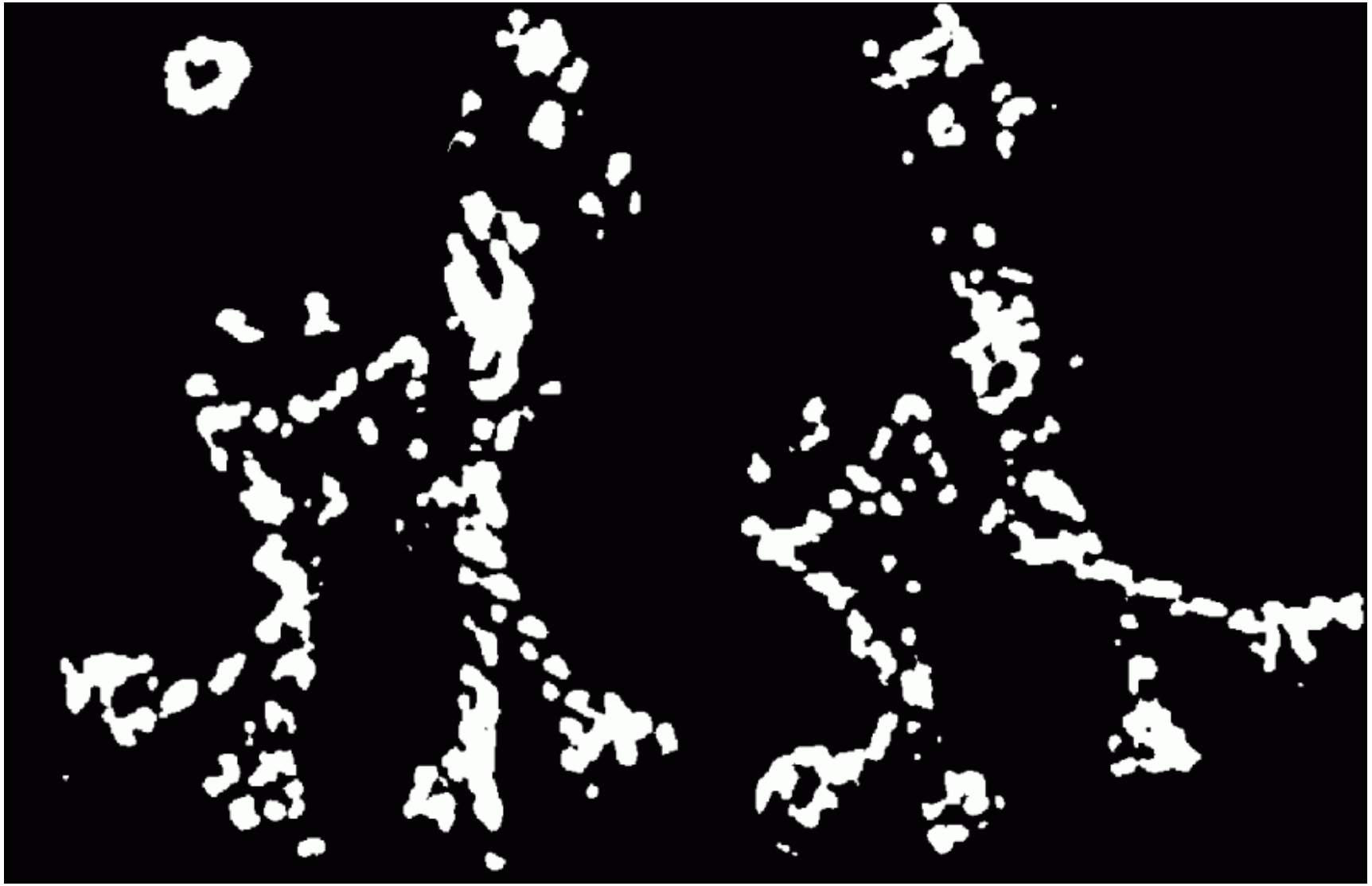
- Compute second moment matrix M in a Gaussian window around each pixel
- Compute corner response function R



Harris Detector: Steps

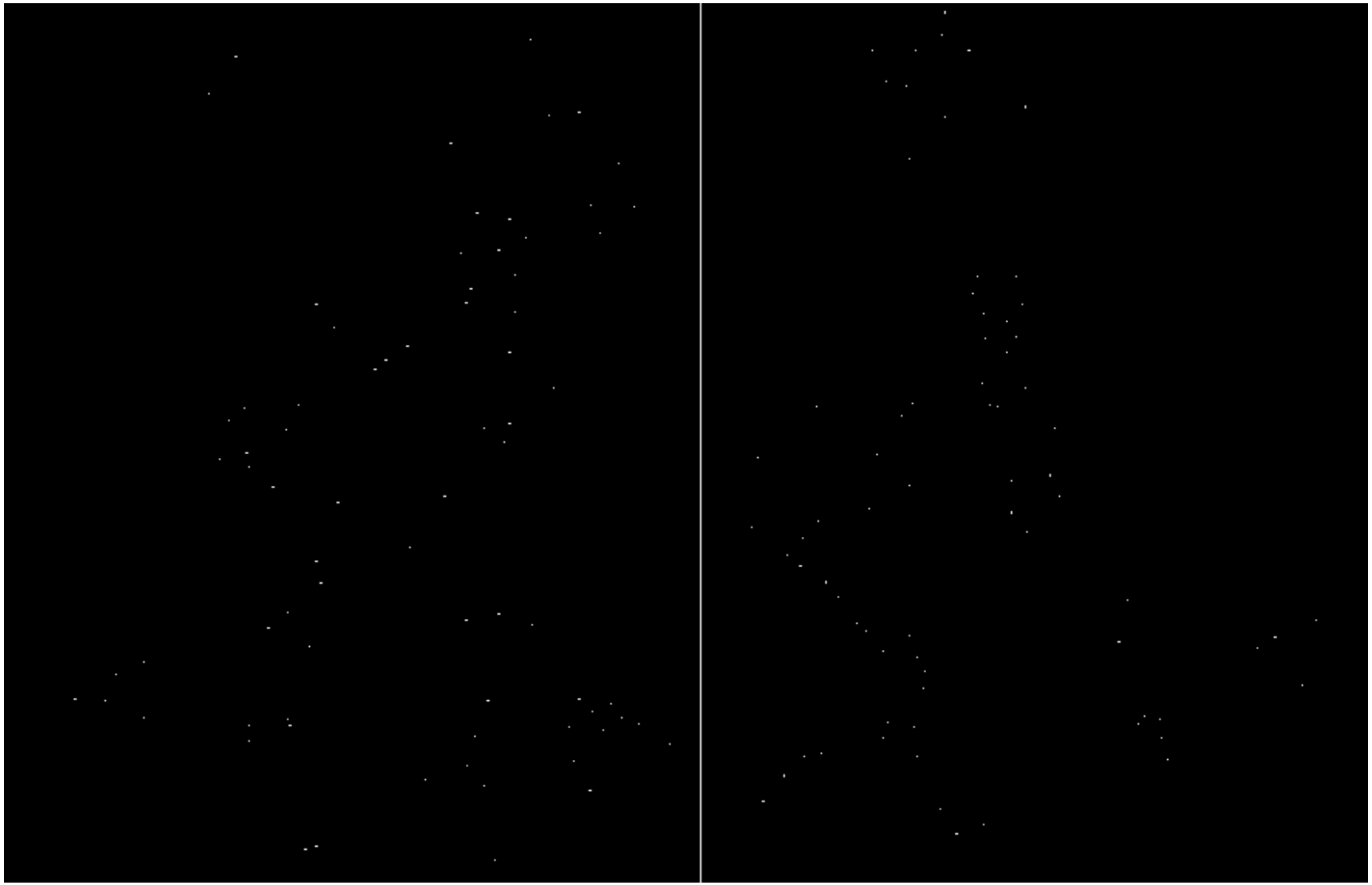
- Find points with large corner response:

$$R > threshold$$



Harris Detector: Steps

- Take only the points of local maxima of R



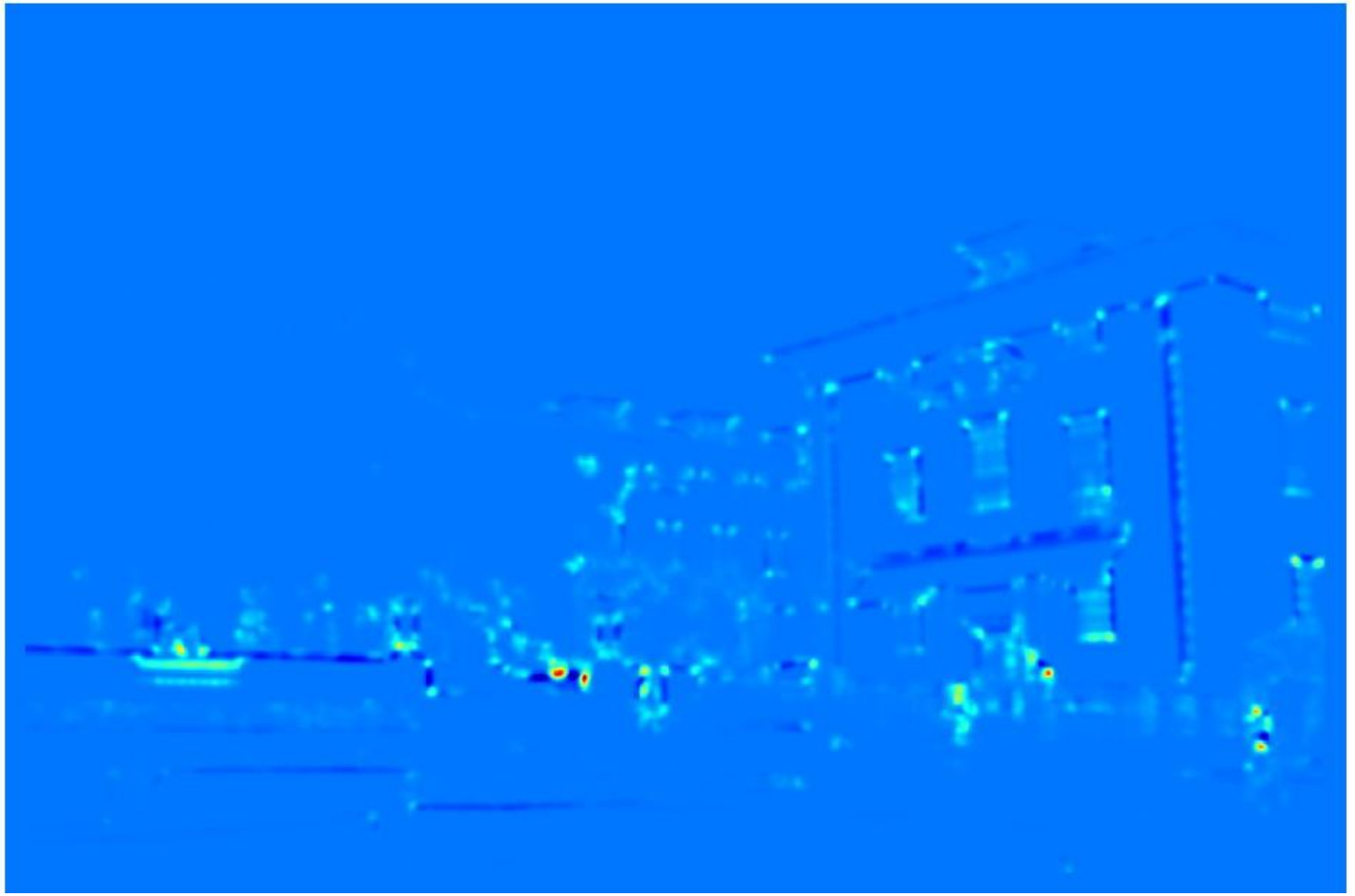
Harris Detector: Steps

Example of Harris application

K. Grauman

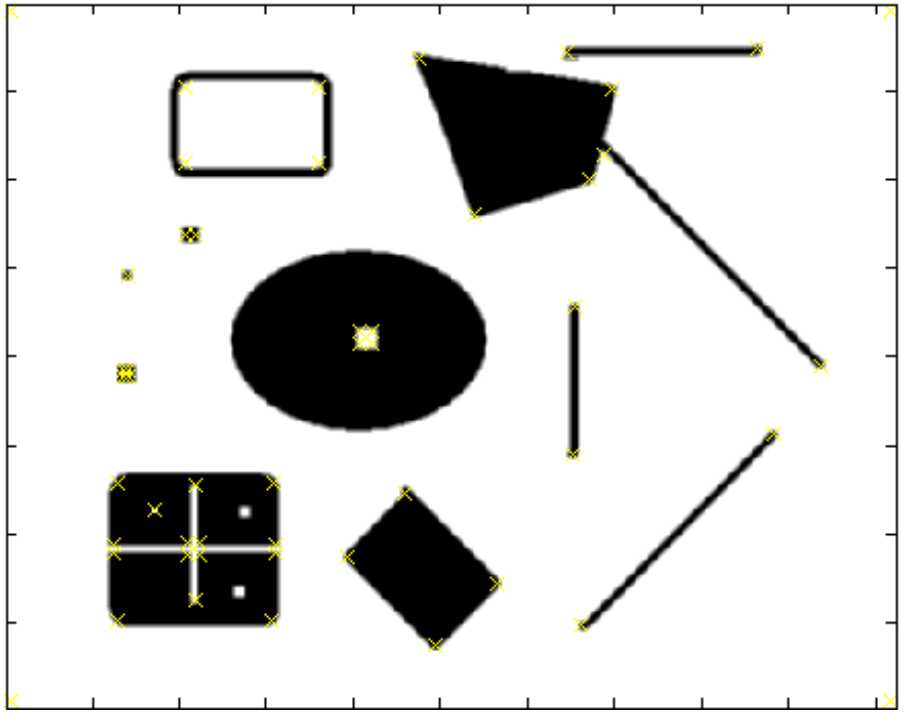
Example of Harris application

- Corner response at every pixel

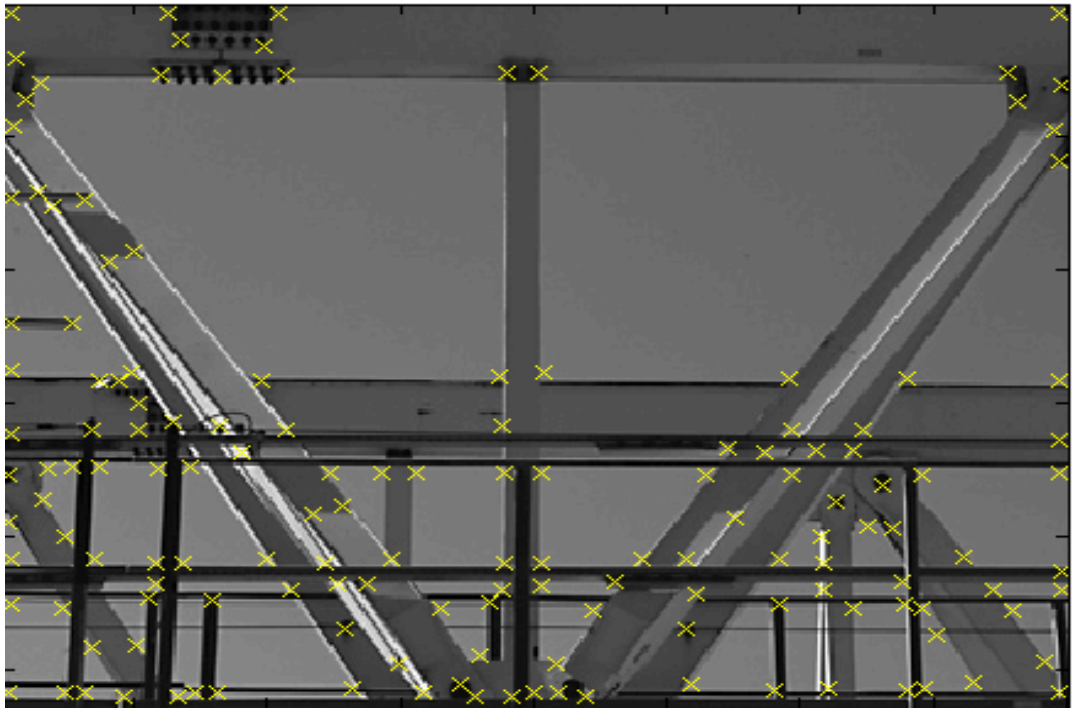


K. Grauman

More Harris responses



Effect: **A very precise corner detector.**



D. Hoiem

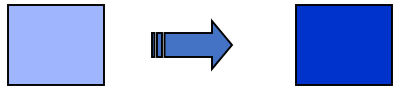
More Harris responses

D. Hoiem

Robustness of corner features

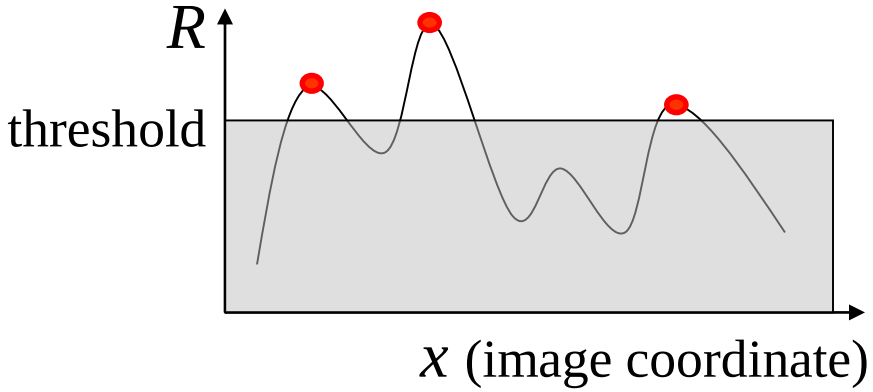
- What happens to corner features when the image undergoes geometric or photometric transformations?
  - Intensity changes
  - Translations
  - Rotation
  - Scaling

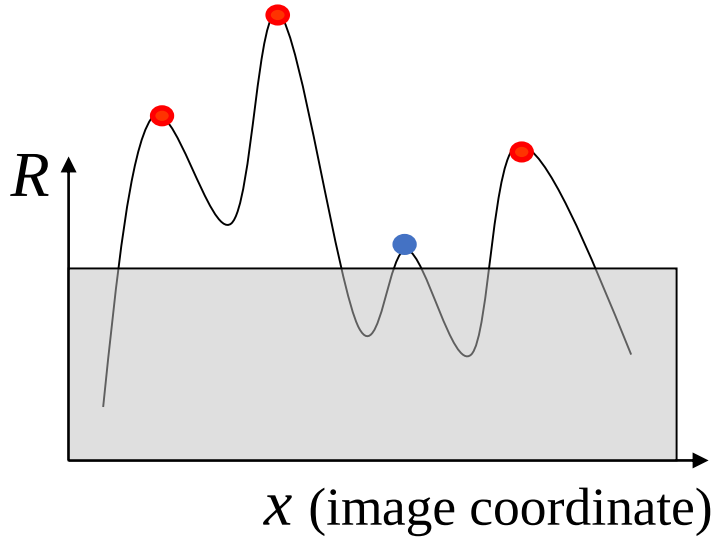
Affine intensity change



$$I \rightarrow a\,I + b$$

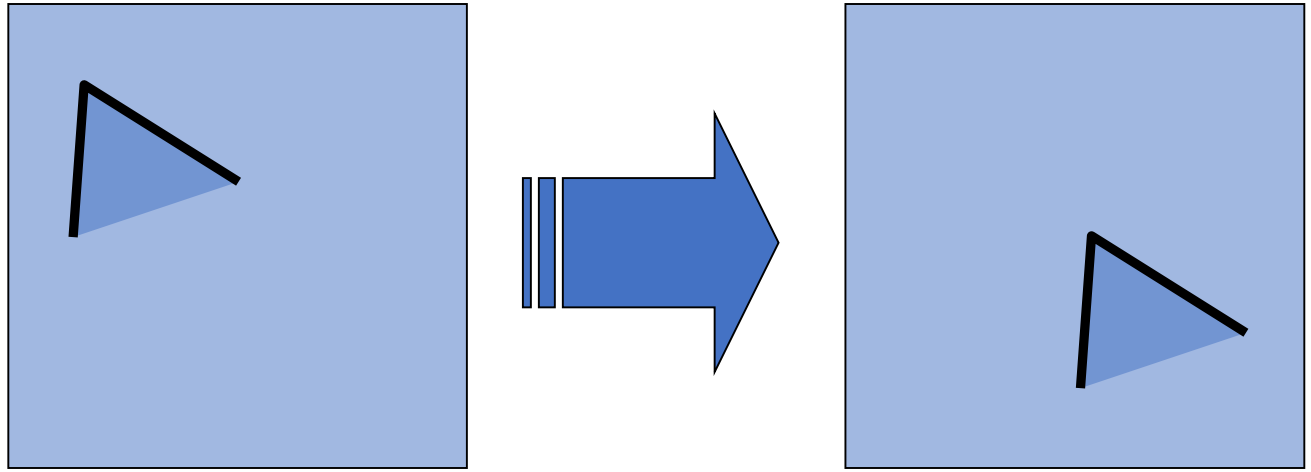
- Only derivatives are used, so invariant to intensity shift $I \rightarrow I + b$
- Intensity scaling: $I \rightarrow a\,I$





Partially invariant to affine intensity change

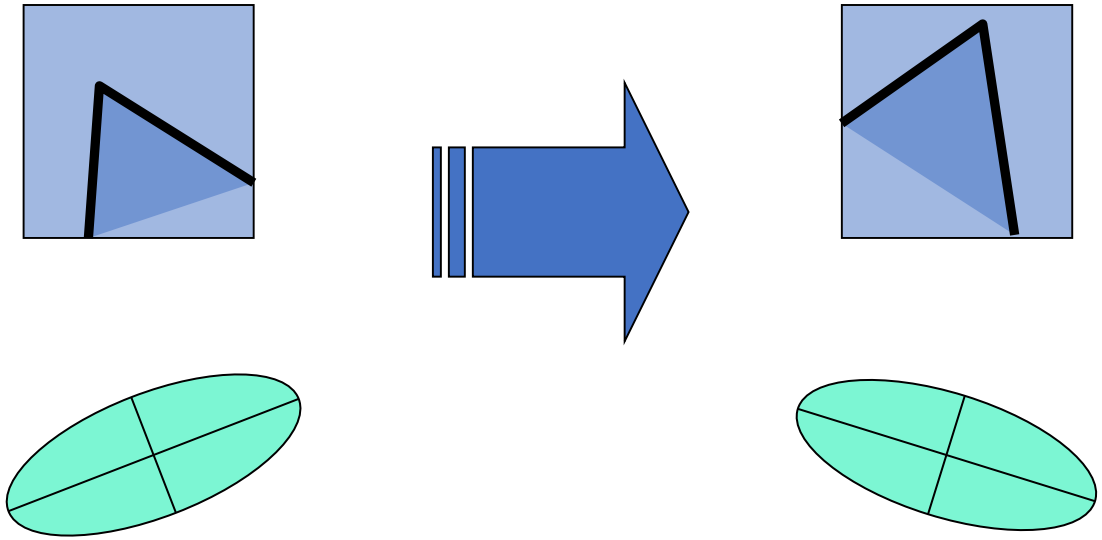
Image translation



- Derivatives and window function are shift-invariant

Corner location is *covariant* w.r.t. translation

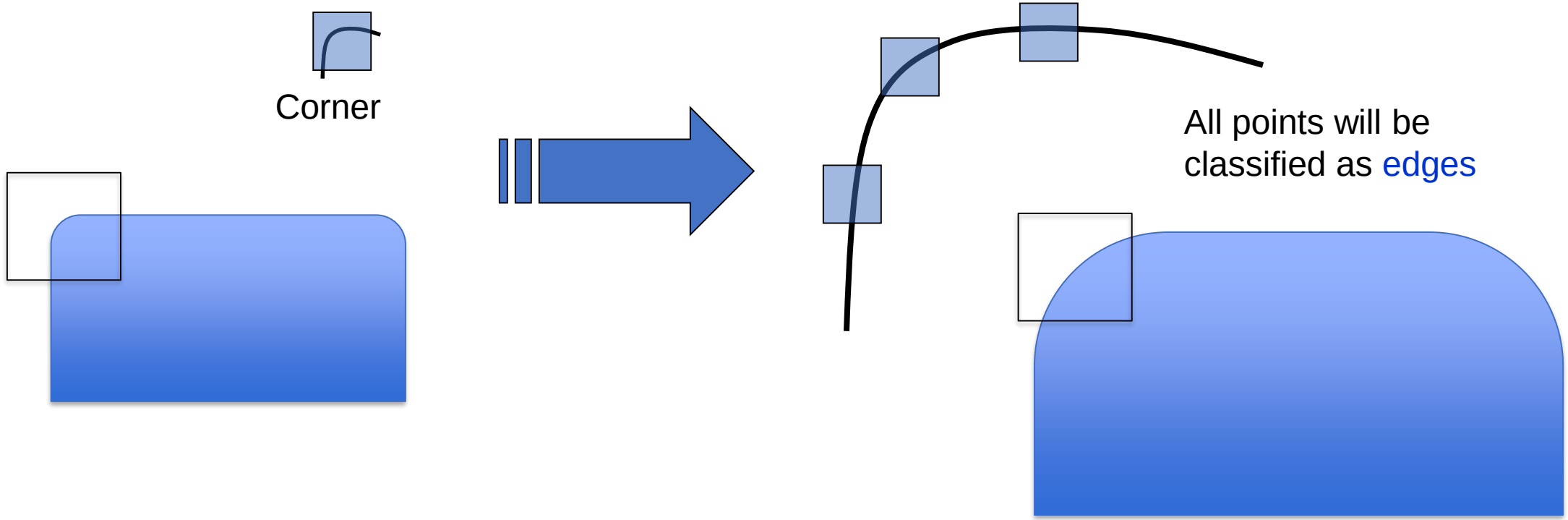
Image rotation



Second moment ellipse rotates but its shape (i.e. eigenvalues) remains the same

Corner location is covariant w.r.t. rotation

Scaling



Corner location is not covariant w.r.t. scaling!

Scale

- Let's look at scale first:

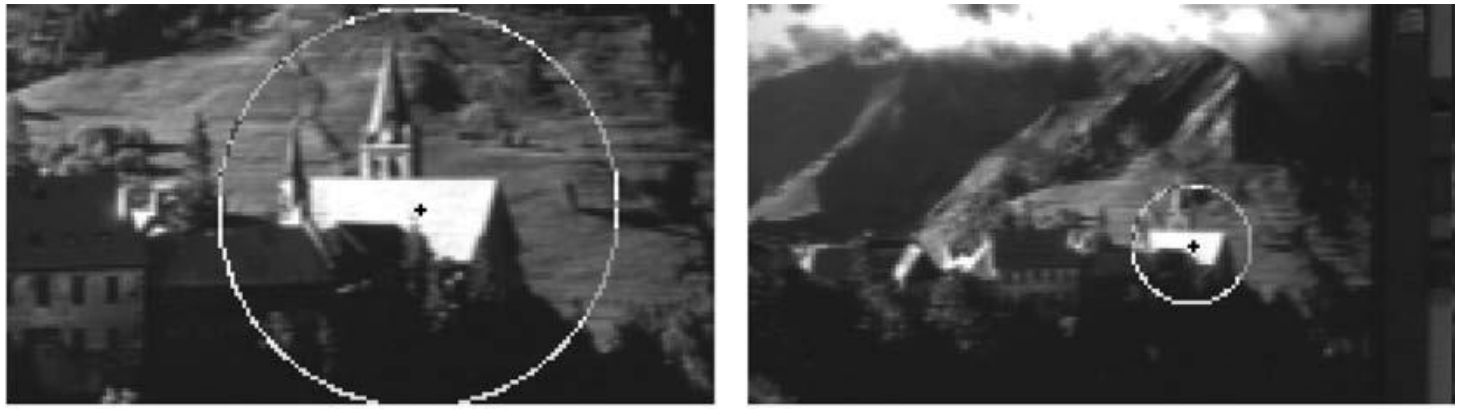
What is the "best" scale?

Does scale matter?

- When detecting corners the ‘scale’ of the window you use can change the corners you detect.

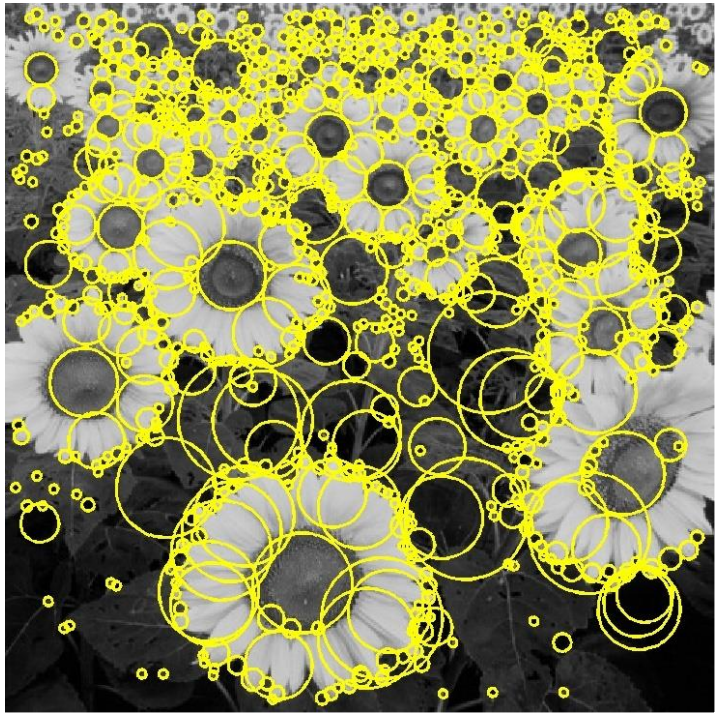
Keypoint detection with scale selection

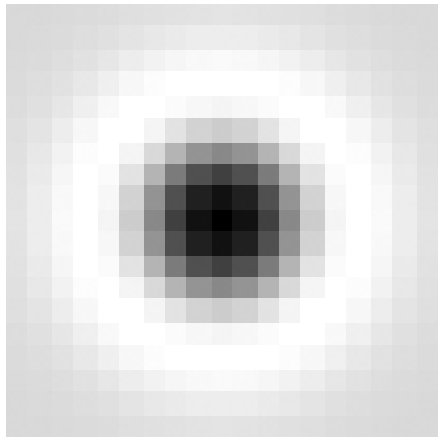
- We want to extract keypoints with characteristic scales that are covariant w.r.t. the image transformation



Basic idea

- Convolve the image with a “blob filter” at multiple scales and look for extrema of filter response in the resulting scale space

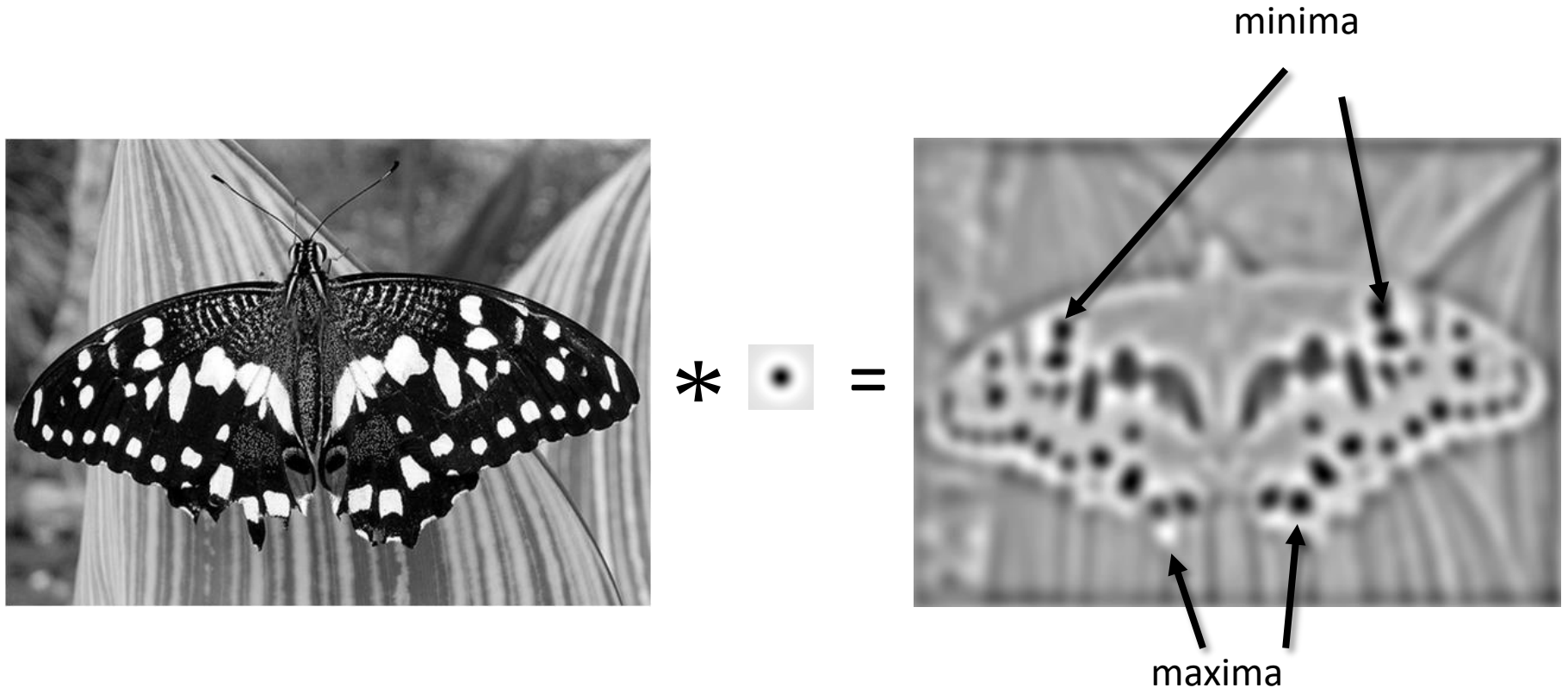




T. Lindeberg, Feature detection with automatic scale selection, *IJCV* 30(2), pp 77-116, 1998

Blob detection

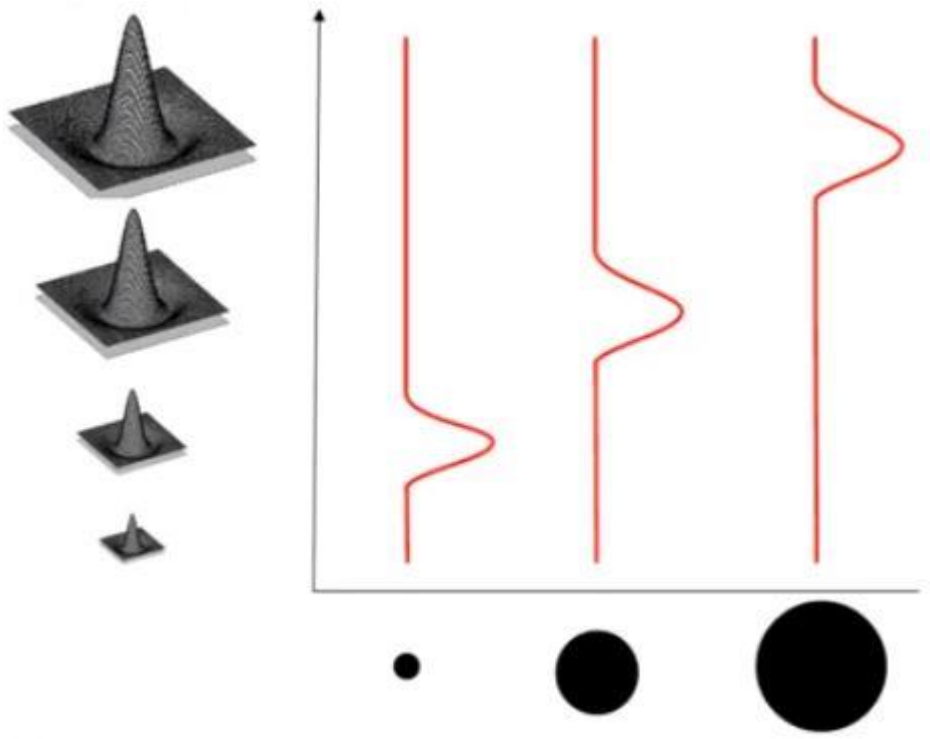
- Find maxima and minima of blob filter response in space and scale

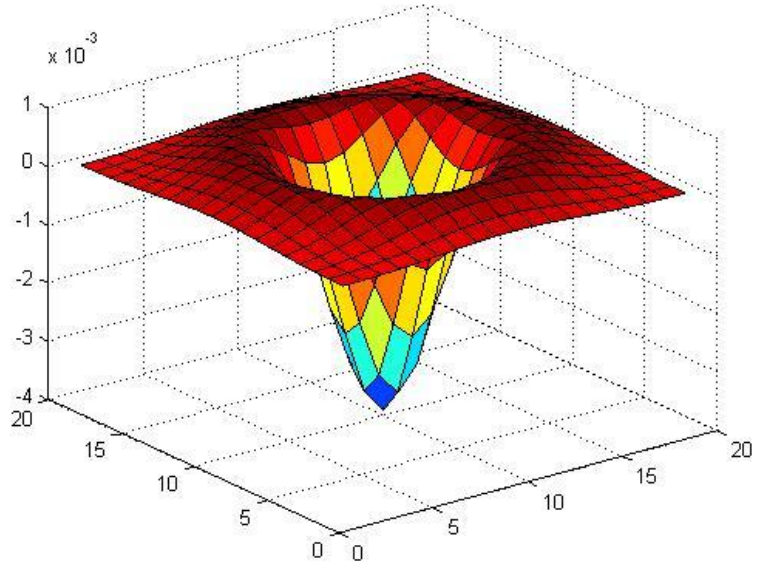


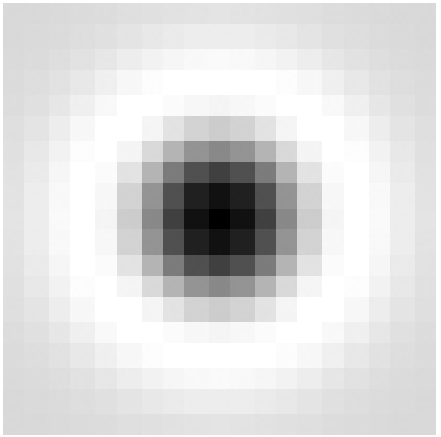
Source: N. Snavely

Blob filter

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



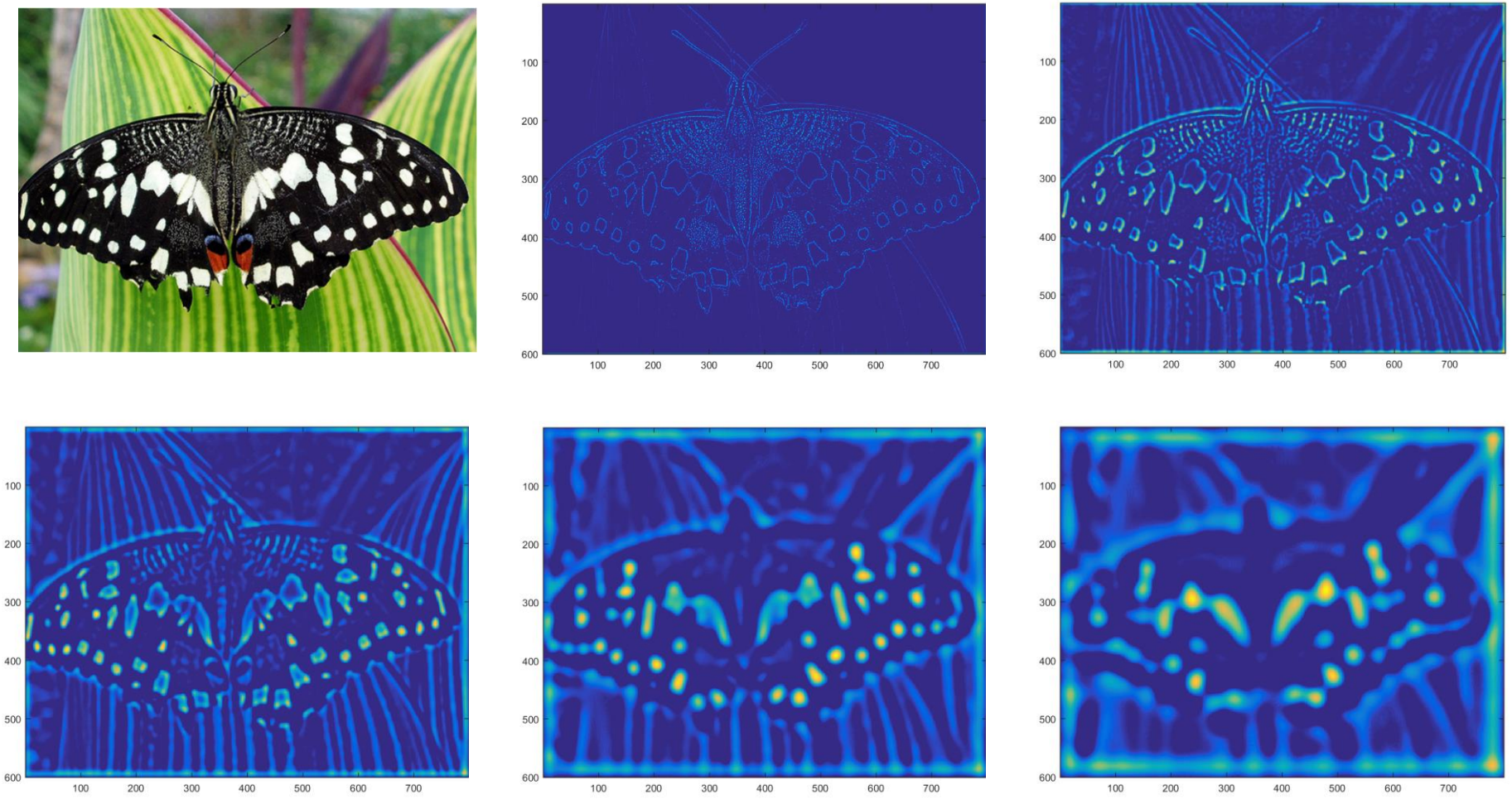




$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

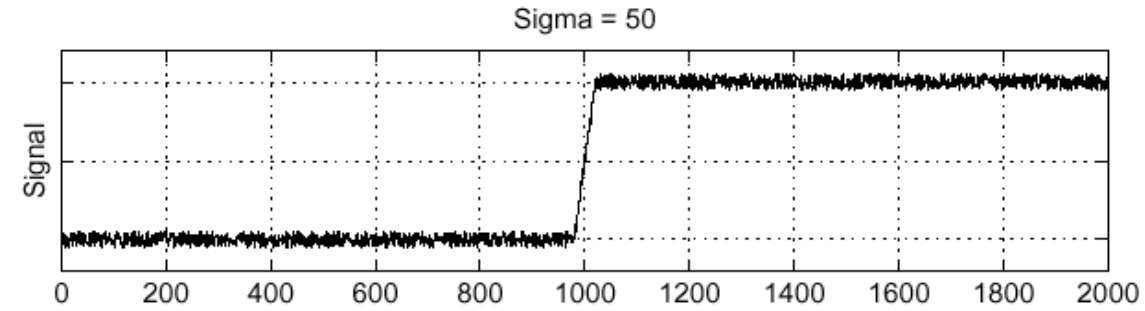
Laplacian pyramid example

- Allows detection of increasingly coarse detail



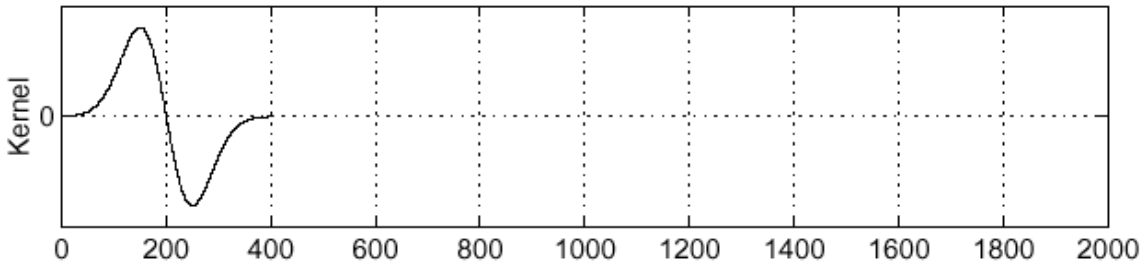
Recall: Edge detection

f



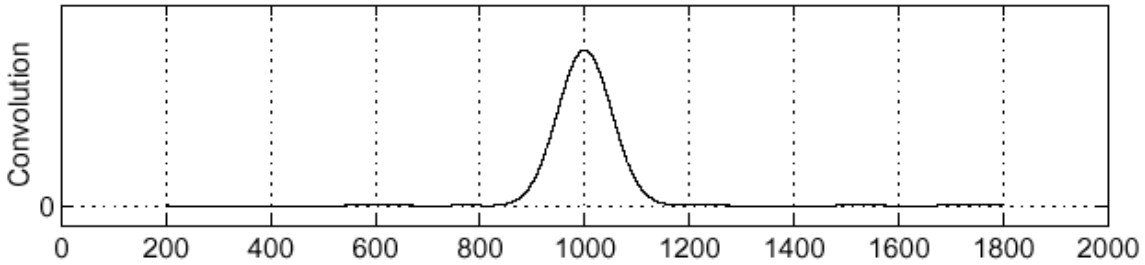
Edge

$\frac{d}{dx} g$



Derivative of Gaussian

$f * \frac{d}{dx} g$

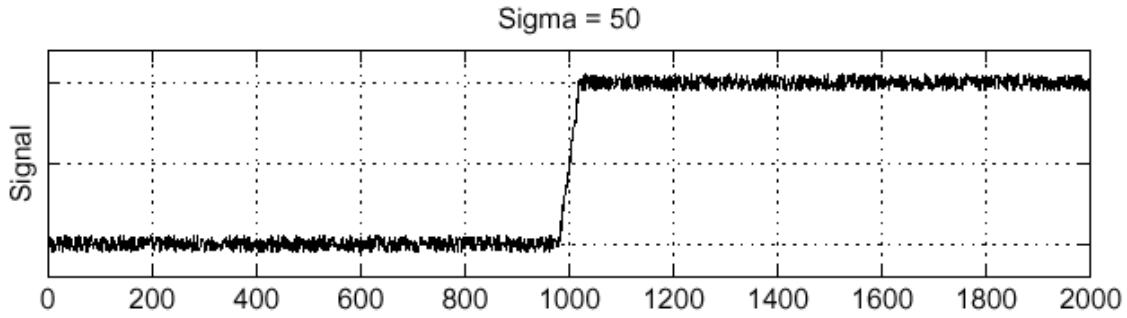


Edge = maximum of derivative

Source: S. Seitz

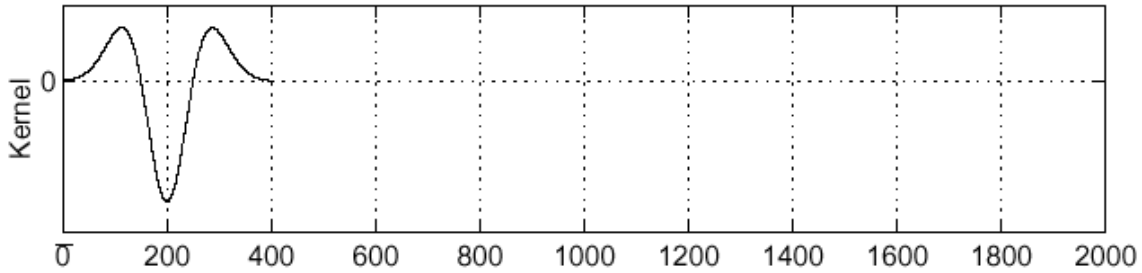
Edge detection, Take 2

f



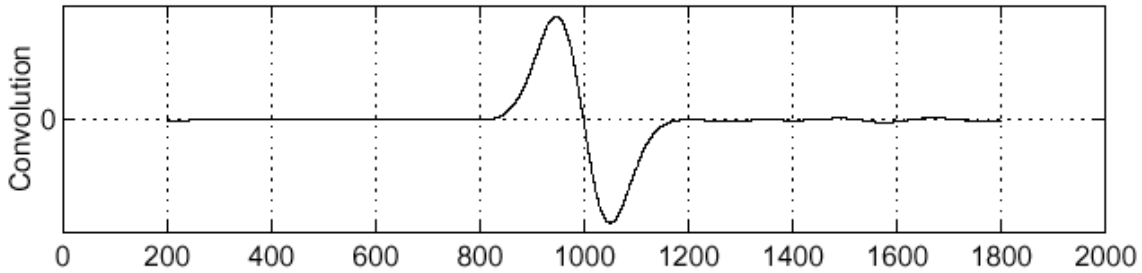
Edge

$\frac{d^2}{dx^2} g$



Second derivative of Gaussian (Laplacian)

$f * \frac{d^2}{dx^2} g$

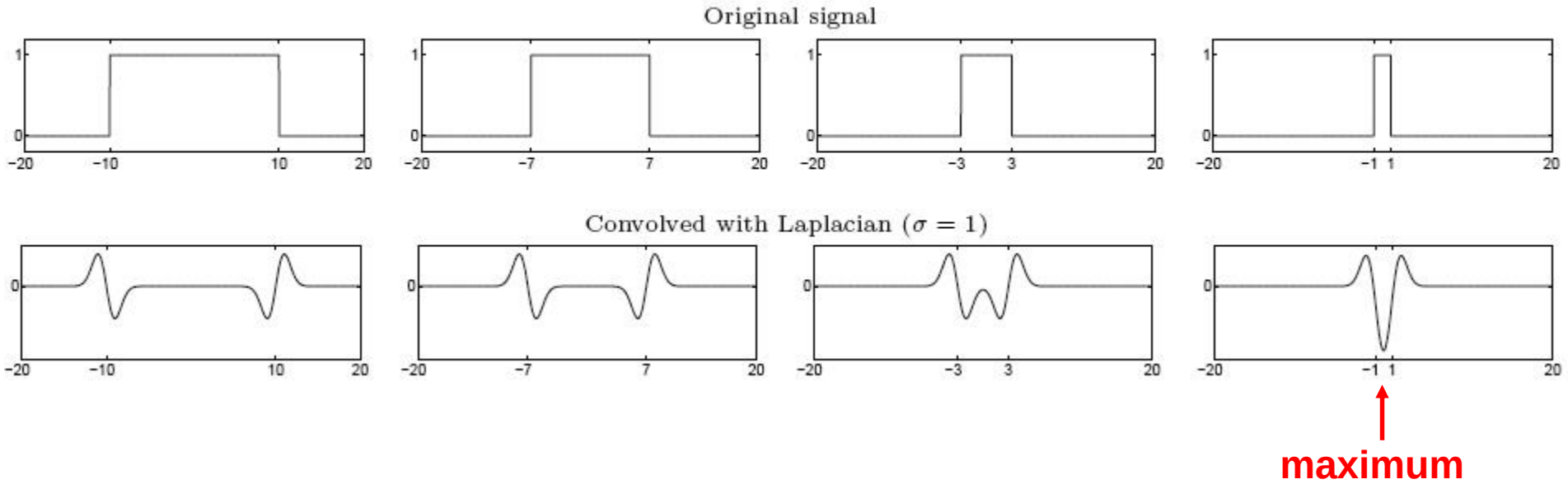


Edge = zero crossing of second derivative

Source: S. Seitz

From edges to blobs

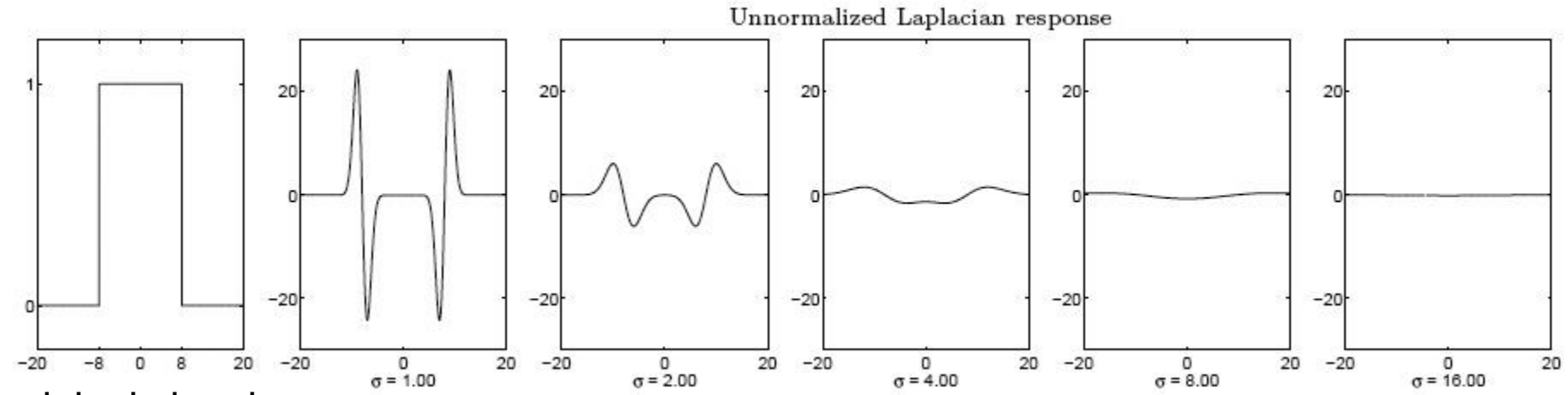
- Edge = ripple
- Blob = superposition of two ripples



- **Spatial selection**: the magnitude of the Laplacian response will achieve a maximum at the center of the blob, provided the scale of the Laplacian is "matched" to the scale of the blob

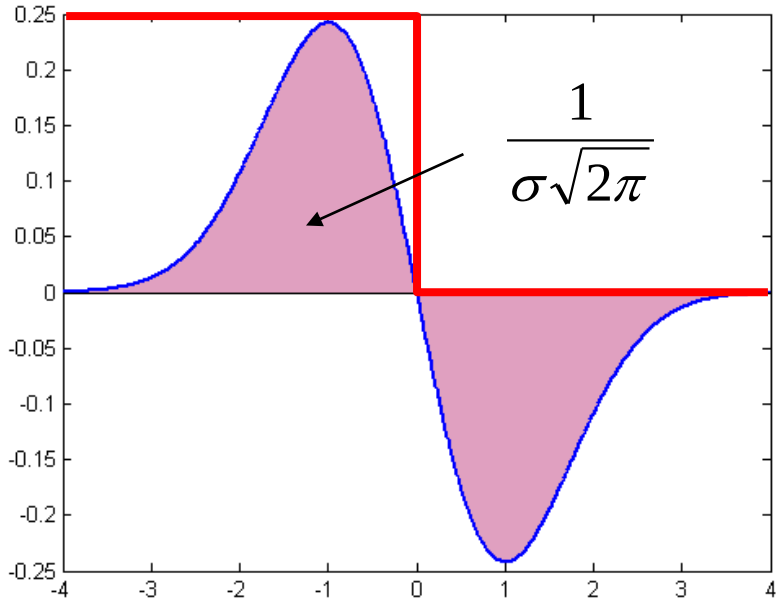
Scale selection

- We want to find the characteristic scale of the blob by convolving it with Laplacians at several scales and looking for the maximum response
- However, Laplacian response decays as scale increases:



Scale normalization

- The response of a derivative of Gaussian filter to a perfect step edge decreases as σ increases:

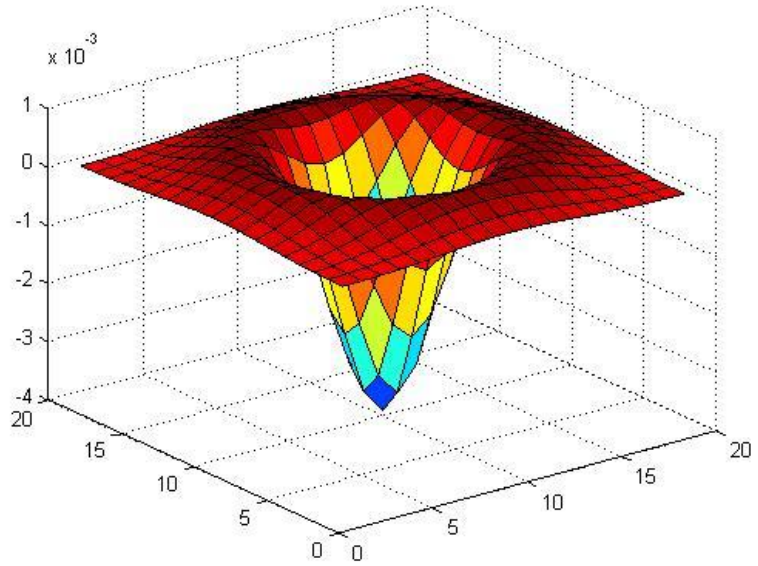


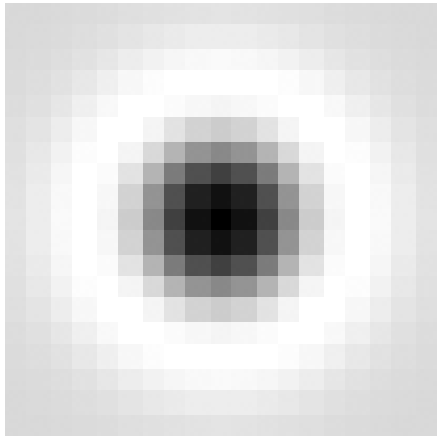
- To keep response the same (scale-invariant), must multiply Gaussian derivative by σ
- Laplacian is the second Gaussian derivative, so it must be multiplied by σ^2

Blob detection in 2D

- Scale-normalized Laplacian of Gaussian:

$$\nabla^2_{\text{norm}} g = \sigma^2 \left(\frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \right)$$

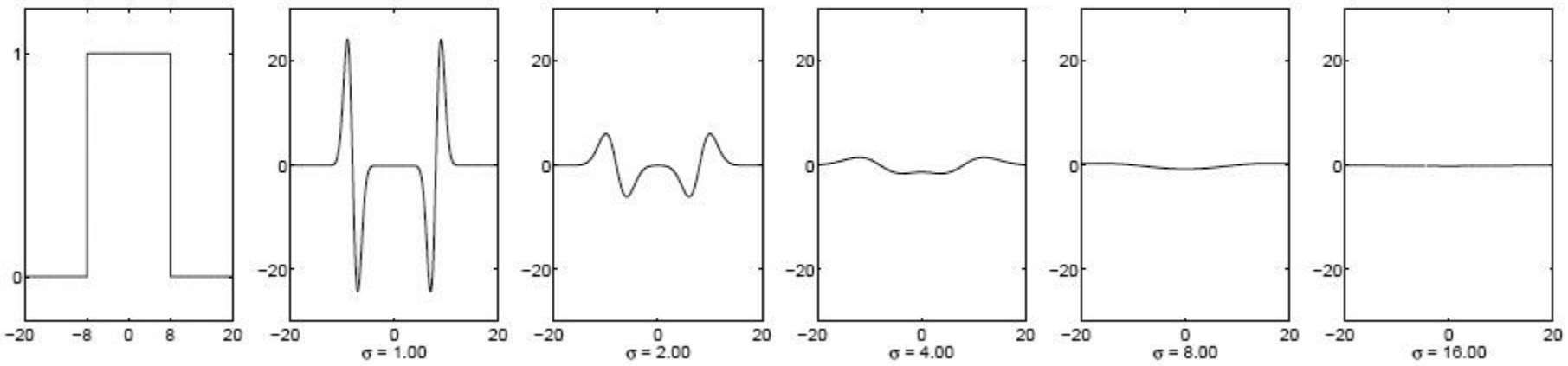




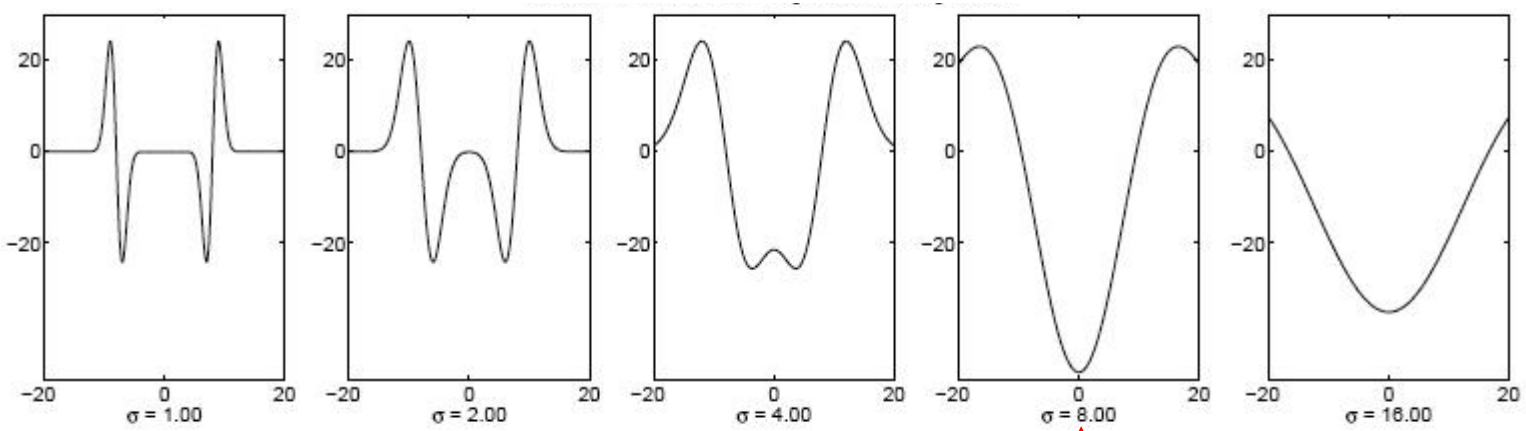
Effect of scale normalization

Original signal

Unnormalized Laplacian response

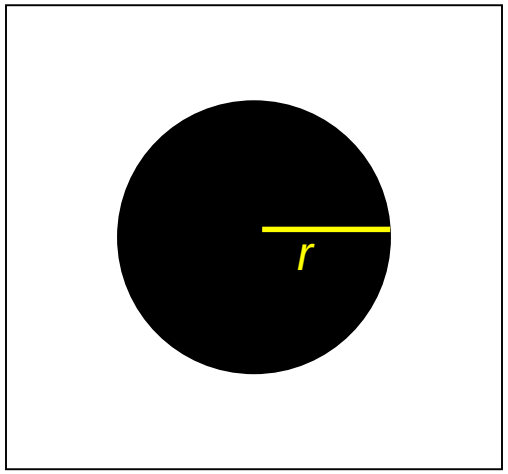


Scale-normalized Laplacian response

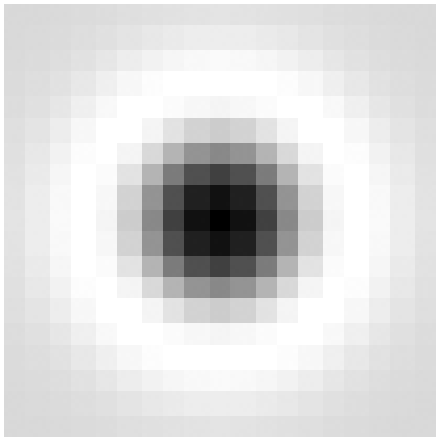


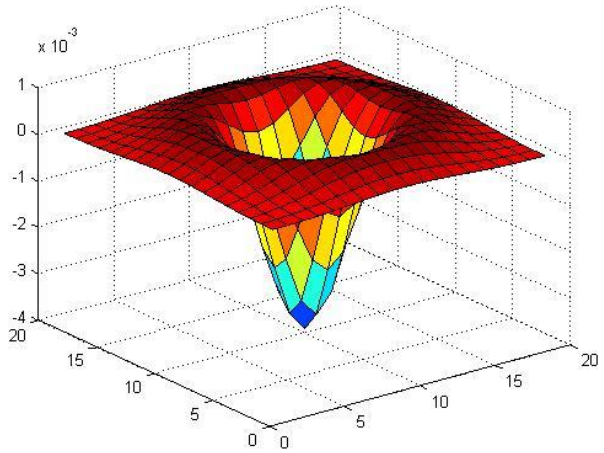
Blob detection in 2D

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



image





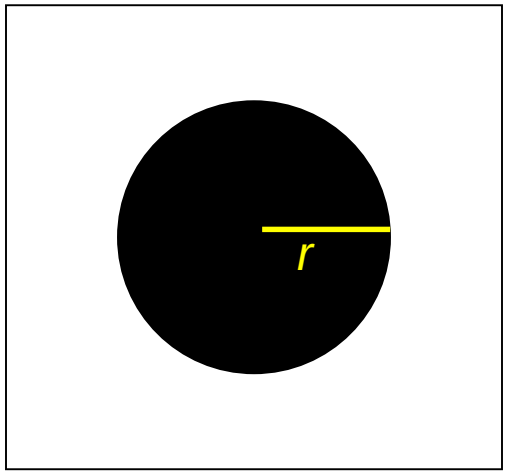
Laplacian

Blob detection in 2D

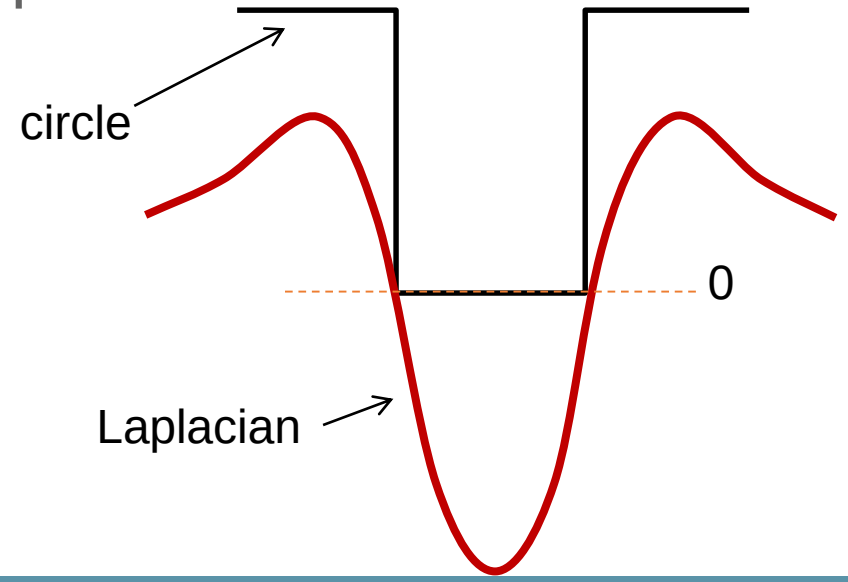
- At what scale does the Laplacian achieve a maximum response to a binary circle of radius r?
- To get maximum response, the zeros of the Laplacian have to be aligned with the circle
- The Laplacian is given by (up to scale):

$$(x^2 + y^2 - 2\sigma^2)\, e^{-(x^2+y^2)/2\sigma^2}$$

- Therefore, the maximum response occurs at $\sigma = r/\sqrt{2}.$



image



Scale-space blob detector: Example

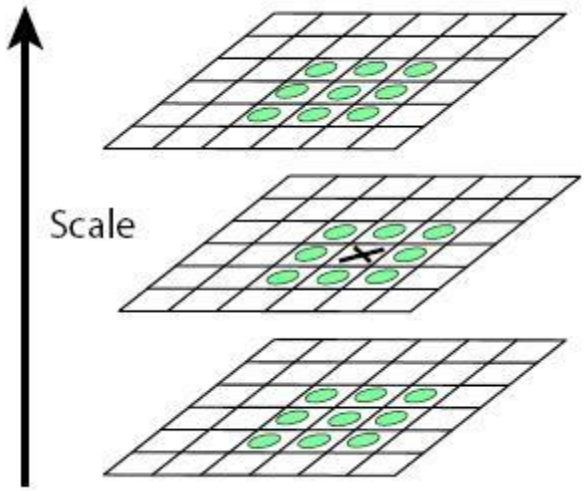
- Convolve image with scale-normalized Laplacian at several scales

Scale-space blob detector: Example

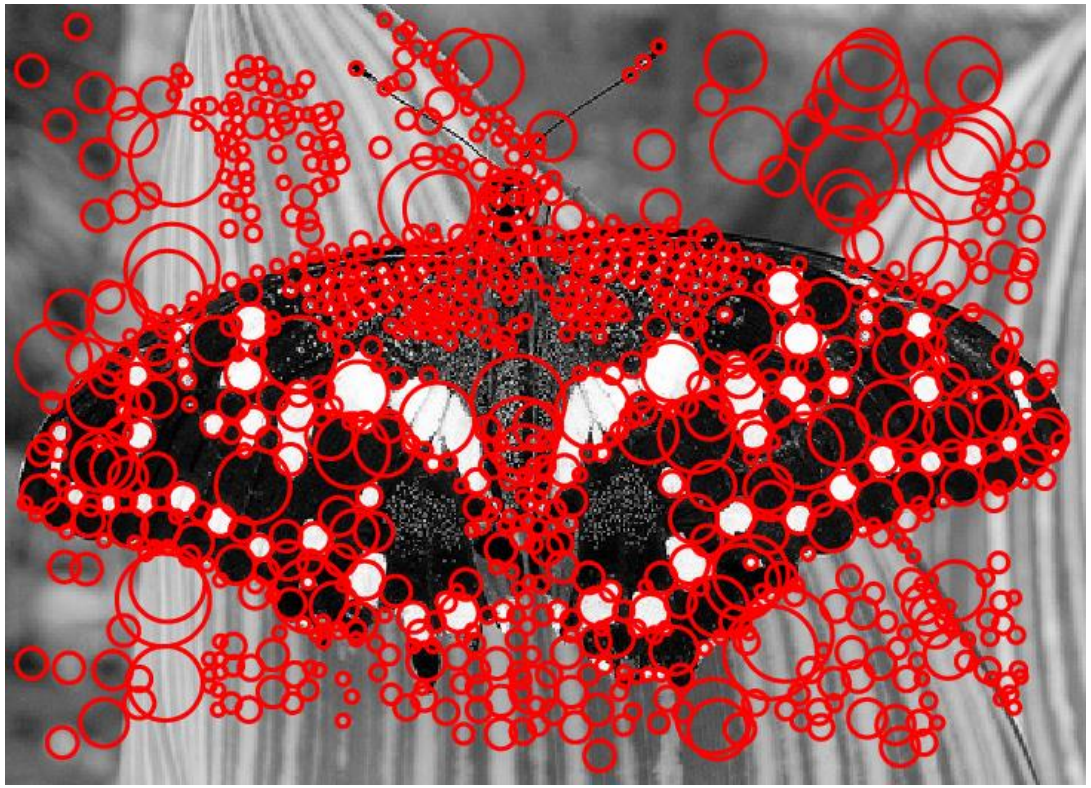
sigma = 11.9912

Scale-space blob detector

- Convolve image with scale-normalized Laplacian at several scales
- Find maxima of squared Laplacian response in scale-space



Scale-space blob detector: Example



Efficient implementation

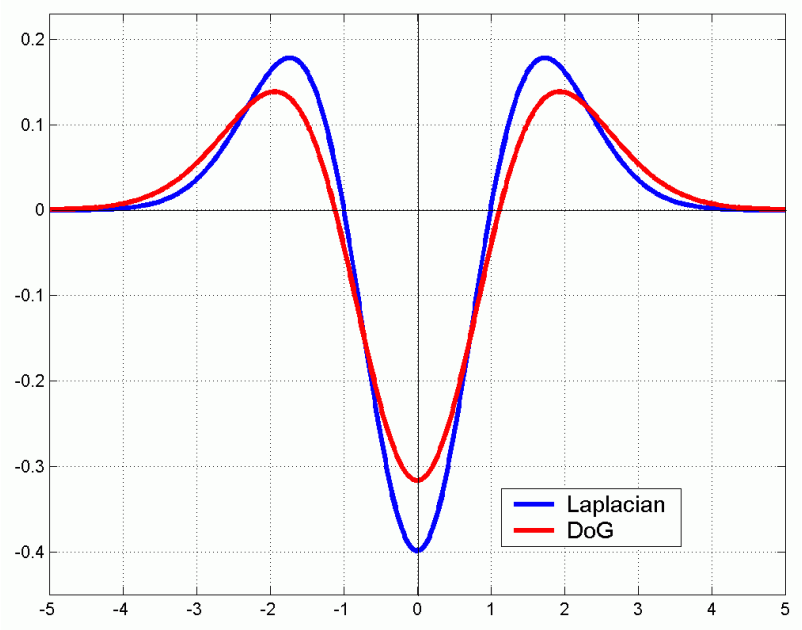
- Approximating the Laplacian with a difference of Gaussians:

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

(Laplacian)

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

(Difference of Gaussians)

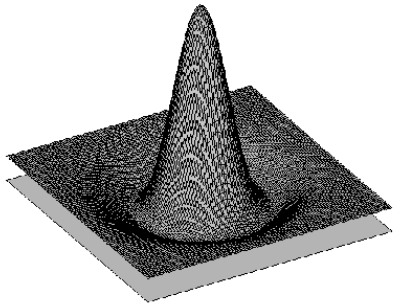


Difference of Gaussians

- Difference of Gaussians (DoG) is a feature enhancement algorithm that involves the subtraction of one blurred version of an original image from another, less blurred version of the original.
- In the simple case of grayscale images, the blurred images are obtained by convolving the original grayscale images with Gaussian kernels having differing standard deviations.
- Blurring an image using a Gaussian kernel suppresses only high-frequency spatial information. Subtracting one image from the other preserves spatial information that lies between the range of frequencies that are preserved in the two blurred images.
- Thus, the difference of Gaussians is a band-pass filter that discards all but a handful of spatial frequencies that are present in the original grayscale image

https://en.wikipedia.org/wiki/Difference_of_Gaussians

Difference-of-Gaussian (DoG)



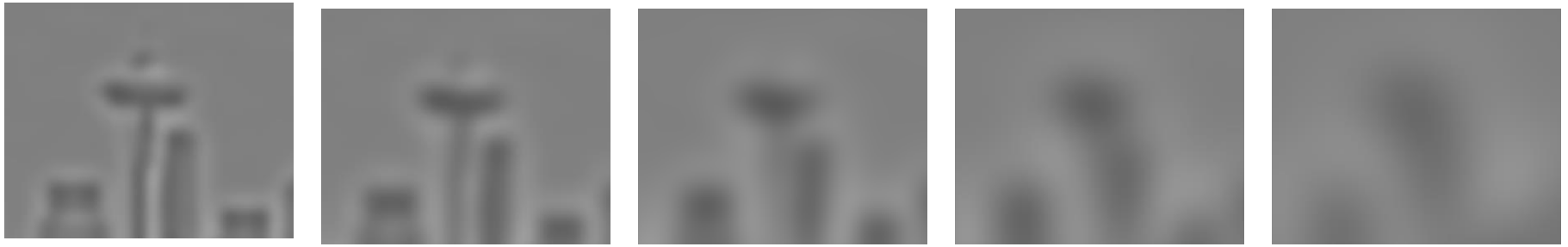
G1 - G2 = DoG

K. Grauman, B. Leibe

DoG example

Take Gaussians at multiple spreads and uses DoGs.

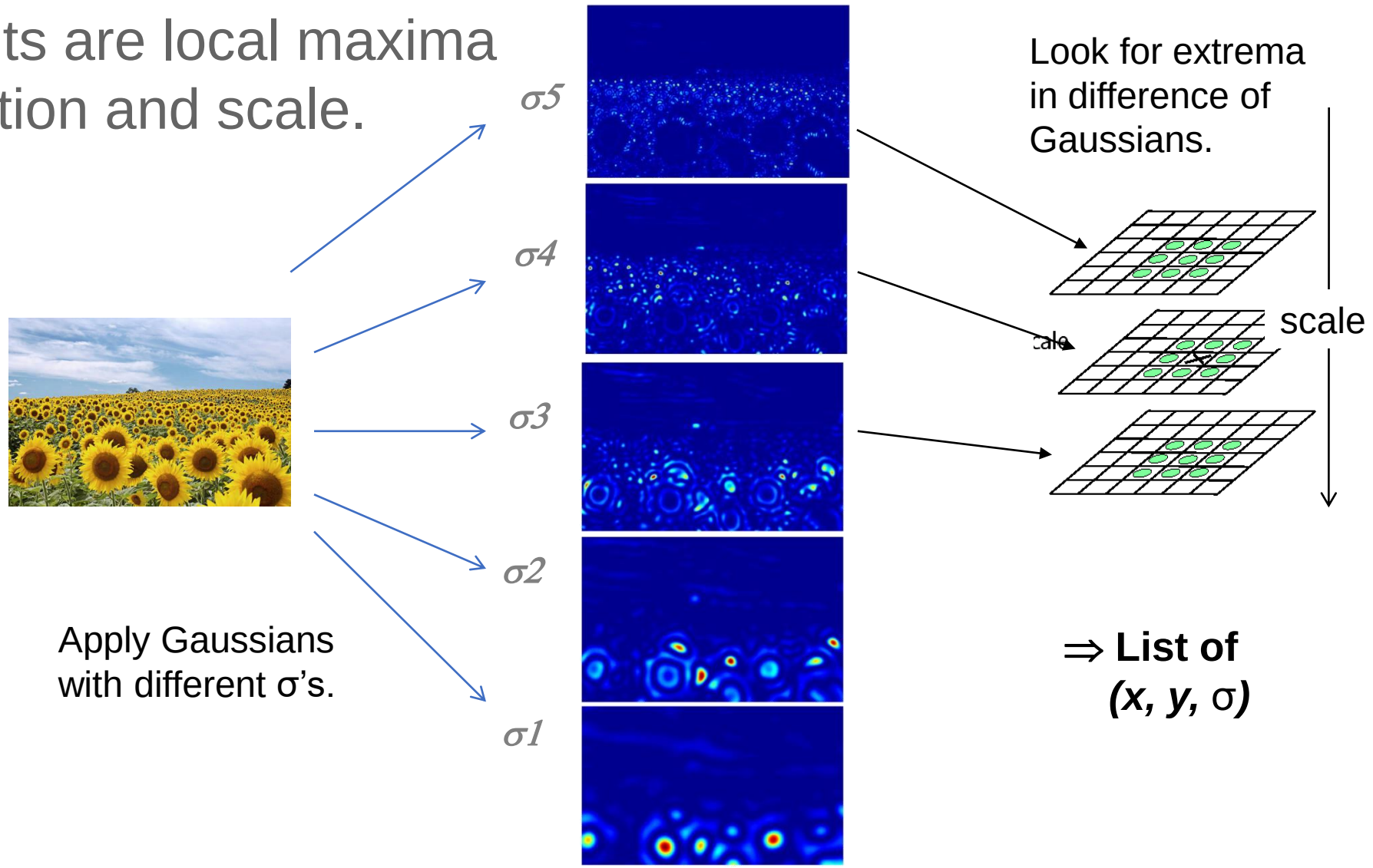
σ = 1



σ = 66

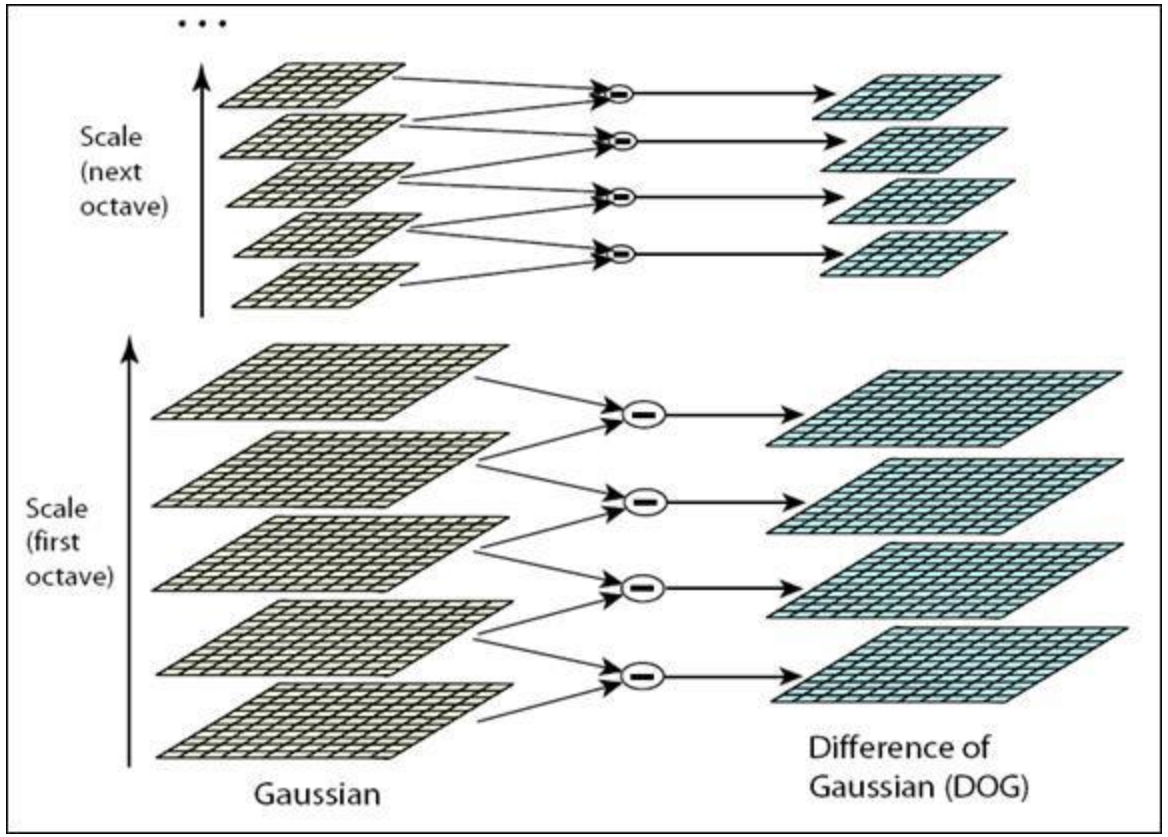
Scale invariant interest points

- Interest points are local maxima in both position and scale.



Scale

In practice the image is downsampled for larger sigmas.



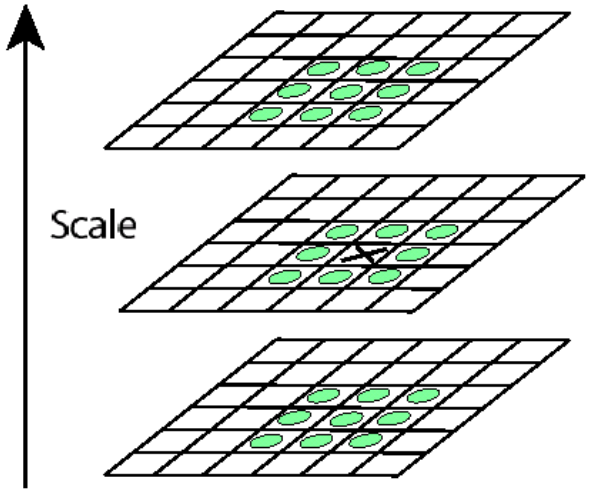
Lowe, 2004.

Key point localization

- Detect maxima and minima of difference-of-Gaussian in scale space

- Each point is compared to its 8 neighbors in the current image and 9 neighbors each in the scales above and below

s+2 difference images.
top and bottom ignored.
s planes searched.



For each max or min found, output is the **location** and the **scale**.

Difference of Gaussians (DoG)

- At different resolutions of kernel size, we see different fine details of the image, In other words, we can capture keypoints at varying scales

Original video

Blurred with a gaussian kernel: k_1

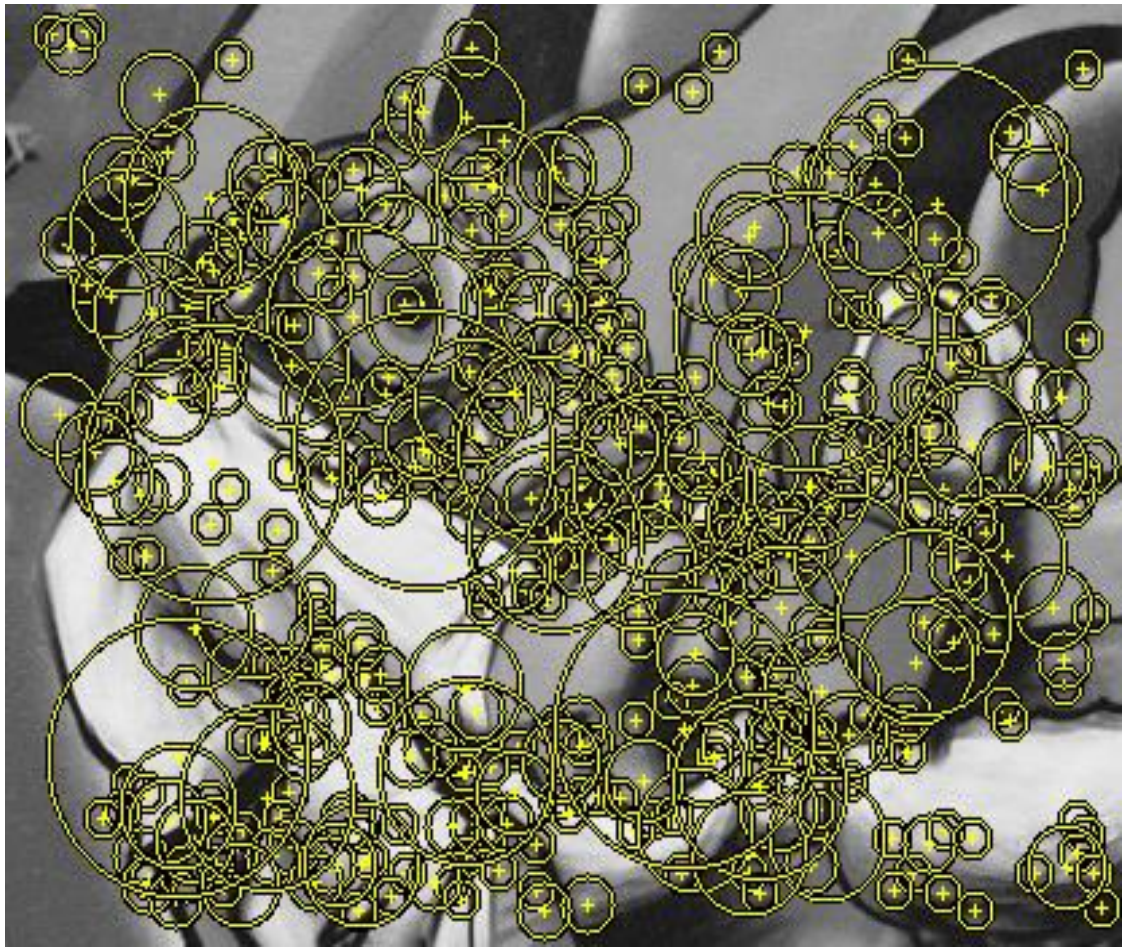
Blurred with a different gaussian kernel: k_2

DoG: $k_1 - k_2$

DoG: $k_1 - k_3$

DoG: $k_1 - k_4$

Results: Difference-of-Gaussian



K. Grauman, B. Leibe

Scale Invariant Detectors: Summary

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

- Methods
  - Harris-Laplacian: Maximize Laplacian over scale, Harris' measure of corner response over the image
  - SIFT: maximize Difference of Gaussian over scale and space

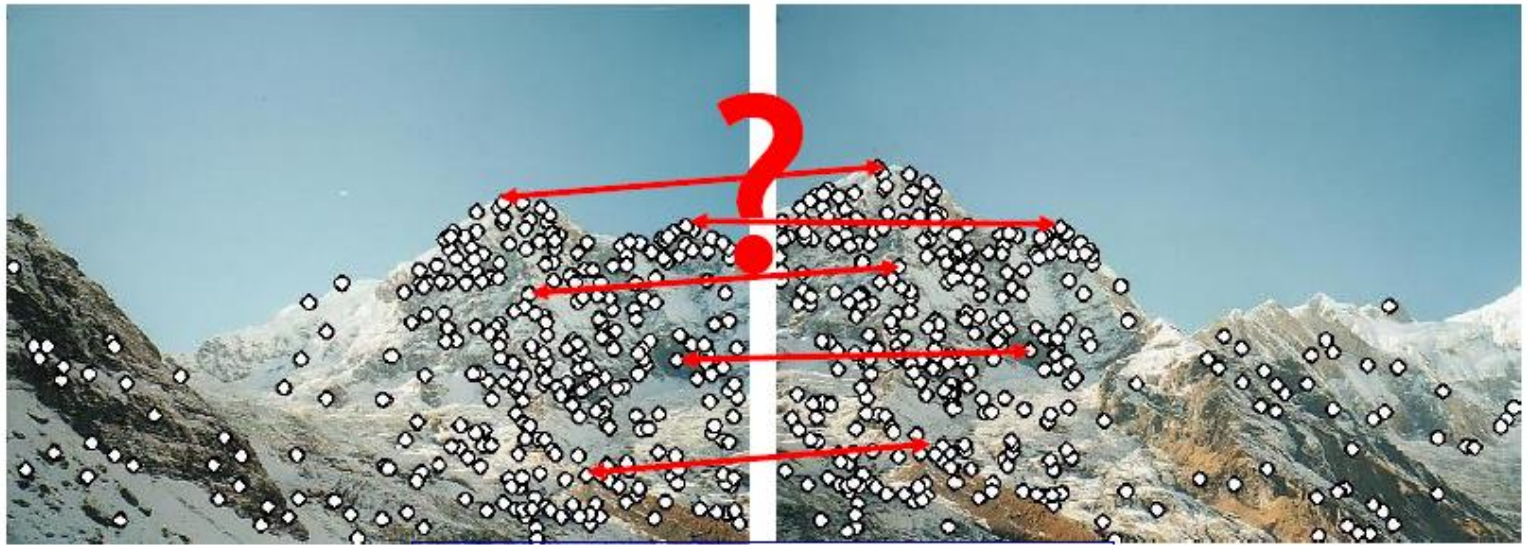
What's next?

- So we can detect keypoints at varying scales. But what can we do with those keypoints?
  - Things we would like to do:
    - Search
      - We would need to find similar key points in other images
    - Panorama
      - Match keypoints from one image to another
    - Etc.

- For all such applications we need a way of describing the keypoints.

Local Descriptors

- We know how to detect points
- Next question:
  - How to describe them for matching?
- Point descriptor should be
  - Invariant
  - Distinctive



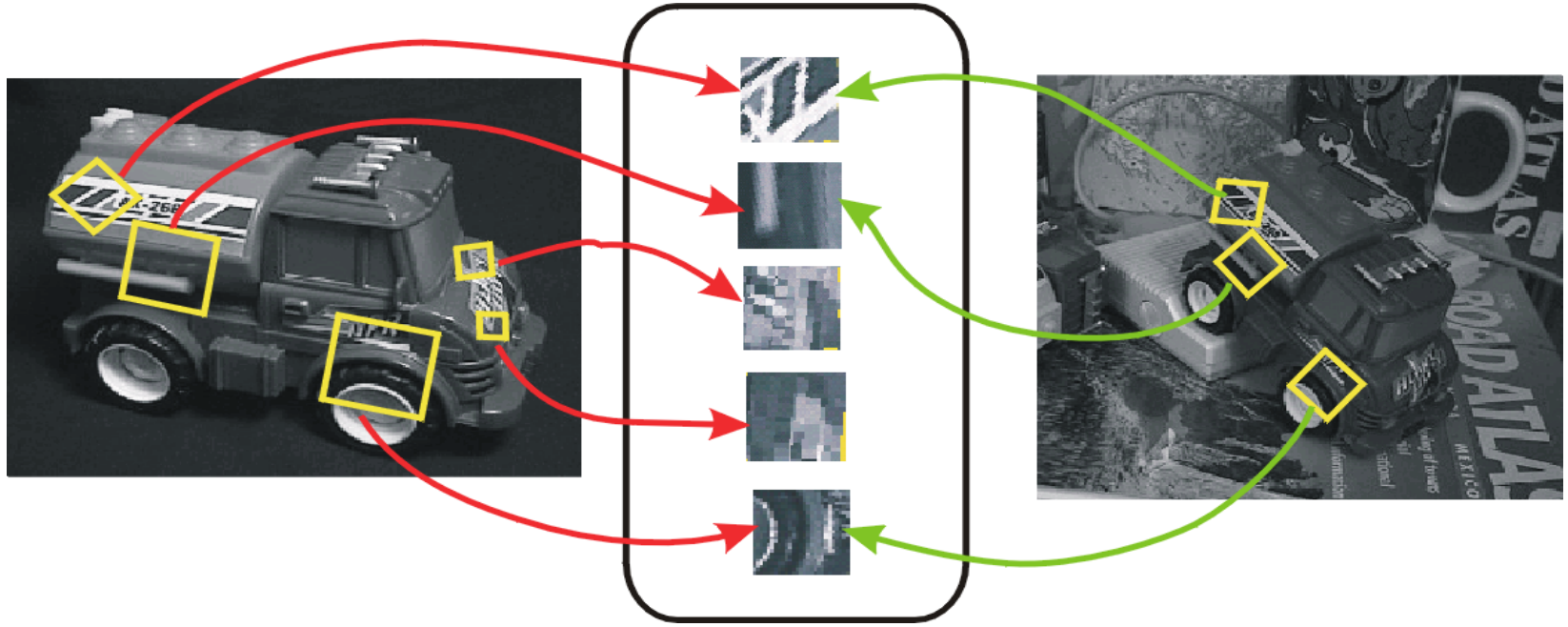
SIFT DESCRIPTORS

David G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints”, International Journal of Computer Vision, 2004

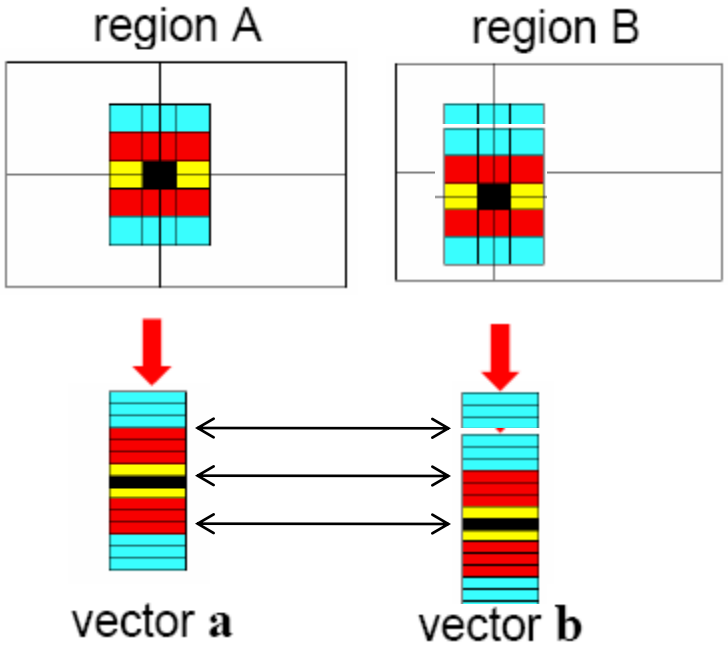
> 55,000 citations

From keypoint detection to feature description

- Detection is covariant:
  - features(transform(image)) = transform(features(image))
- Description is invariant:
  - features(transform(image)) = features(image)



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.

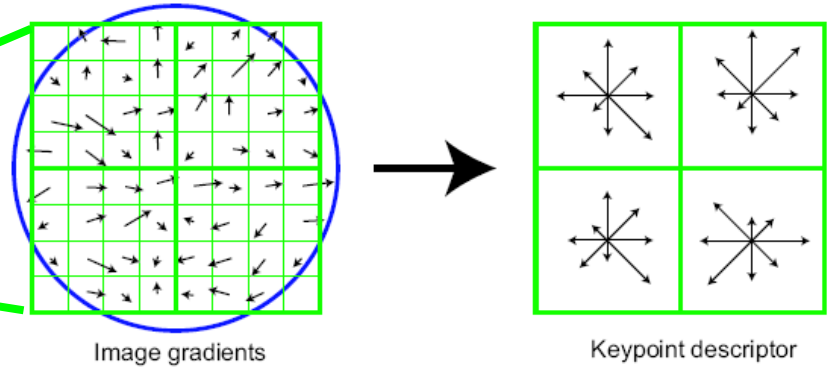
But this is very sensitive to even small shifts, rotations.

Credit: Linda Shapiro

Scale Invariant Feature Transform (SIFT)

- Stands for scale invariant feature transform
- Patented by university of British Columbia
- Similar to the one used in primate visual system (human, ape, monkey, etc.)
- Goal
  - Extracting distinctive invariant features
    - Correctly matched against a large database of features from many images
  - Invariance to image scale and rotation
  - Robustness to
    - Affine distortion
    - Change in 3D viewpoint

Scale-Invariant Feature Transform (SIFT) descriptor



Histogram of oriented gradients

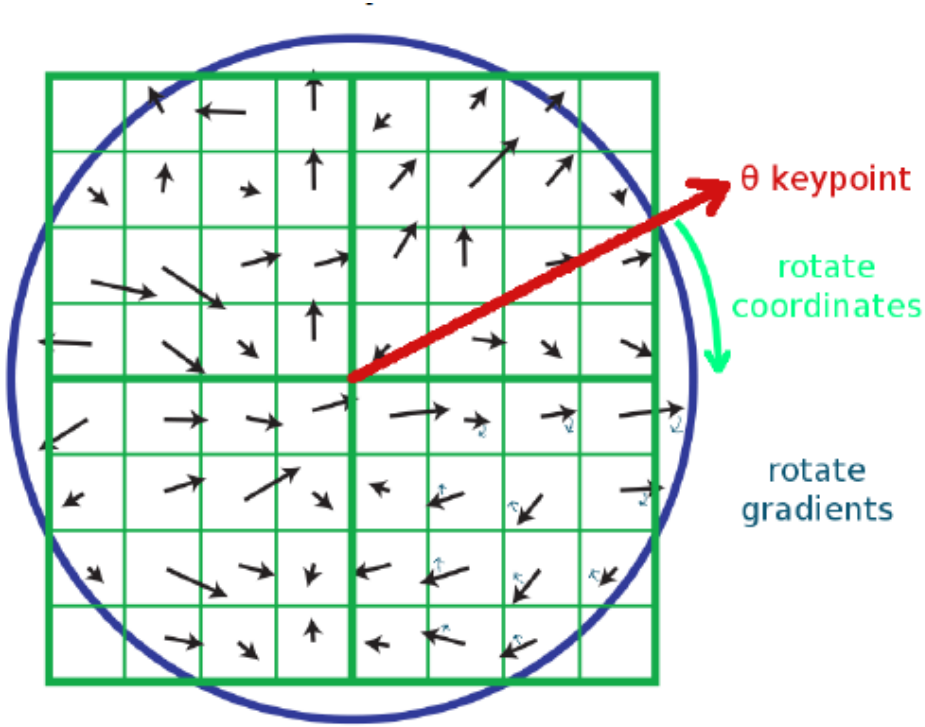
- **Captures important texture information**
- **Robust to small translations / affine deformations**

[Lowe, ICCV 1999]

K. Grauman, B. Leibe

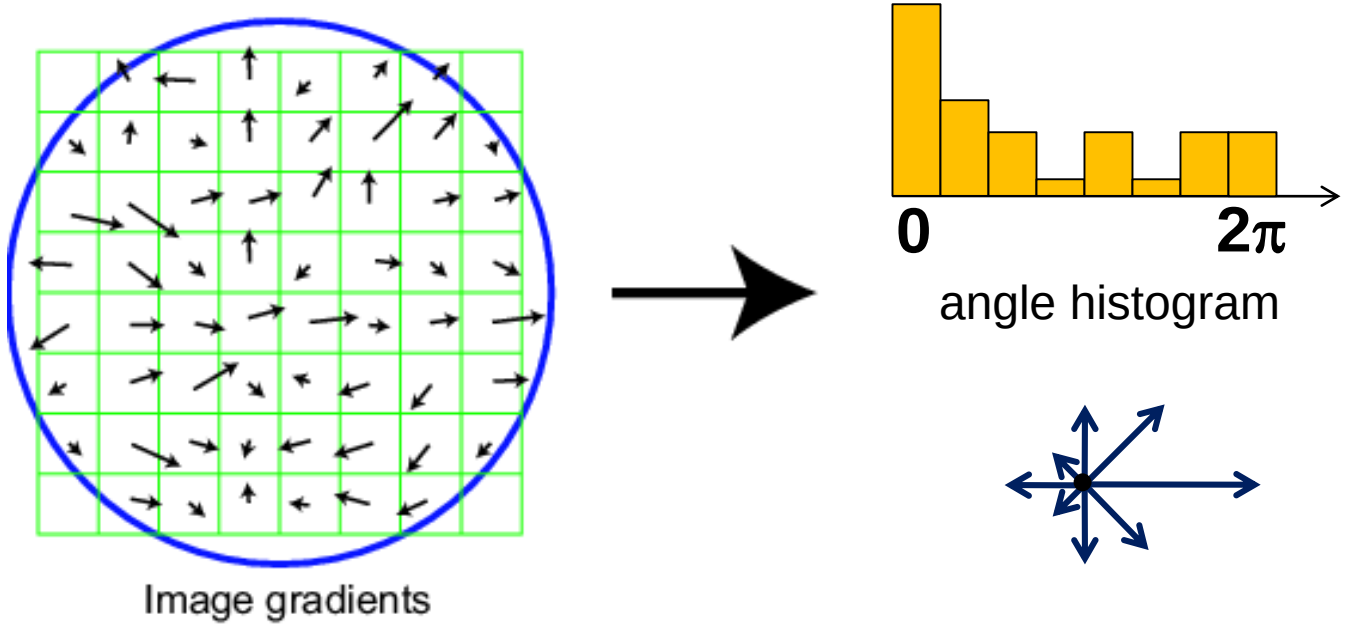
Scale Invariant Feature Transform

- Use the blurred image associated with the keypoints scale
- Take image gradients over the keypoint neighbourhood
- To become rotation invariant, rotate the gradient directions and locations by (-keypoint orientation)
  - Now we have cancelled out rotation and have gradients expressed at locations relative to keypoint orieantation θ



Scale Invariant Feature Transform

- Basic idea:
  - Take 16x16 square window around detected feature
  - Compute gradient orientation for each pixel
  - Create histogram over edge orientations weighted by magnitude
  - That's your feature descriptor!



Adapted from L. Zitnick, D. Lowe

Scale Invariant Feature Transform

- Full version
  - Divide the 16x16 window into a 4x4 grid of cells
  - Quantize the gradient orientations i.e. snap each gradient to one of 8 angles
  - Each gradient contributes not just 1, but magnitude(gradient) to the histogram, i.e. stronger gradients contribute more
  - 16 (4x4) cells * 8 orientations = 128 dimensional descriptor for each detected feature

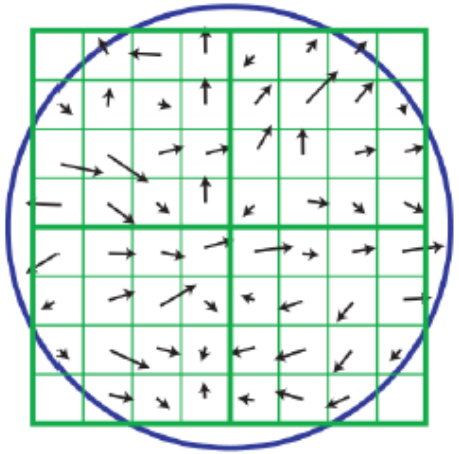
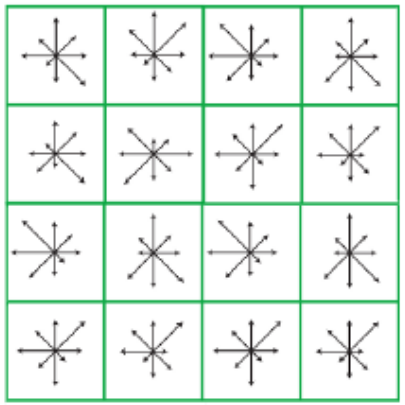
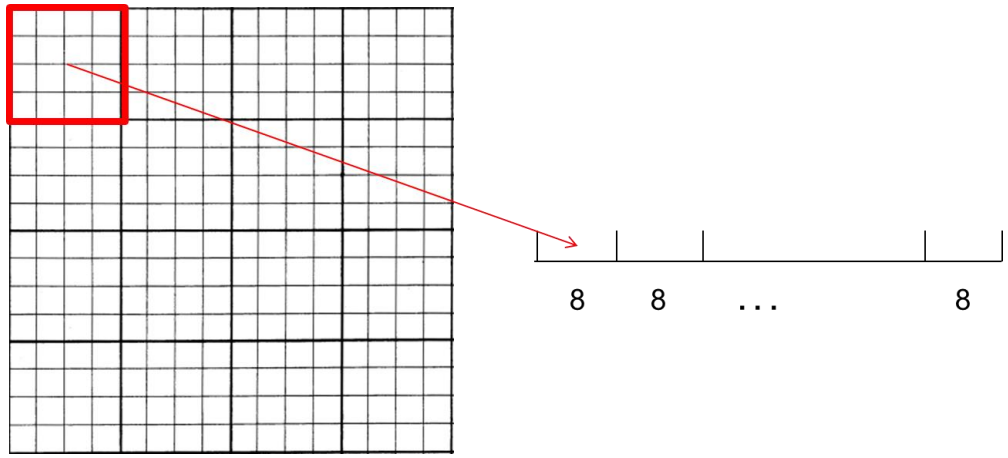


Image gradients



Keypoint descriptor



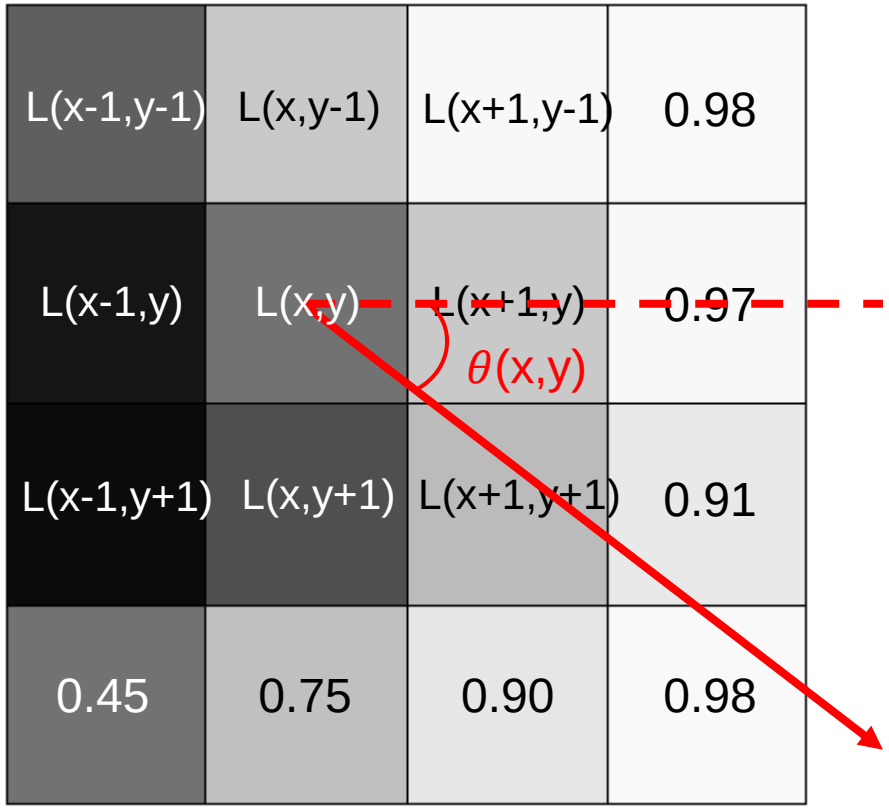
Adapted from L. Zitnick, D. Lowe

Numeric Example

0.37	0.79	0.97	0.98
0.08	0.45	0.79	0.97
0.04	0.31	0.73	0.91
0.45	0.75	0.90	0.98

by Yao Lu

Numeric Example

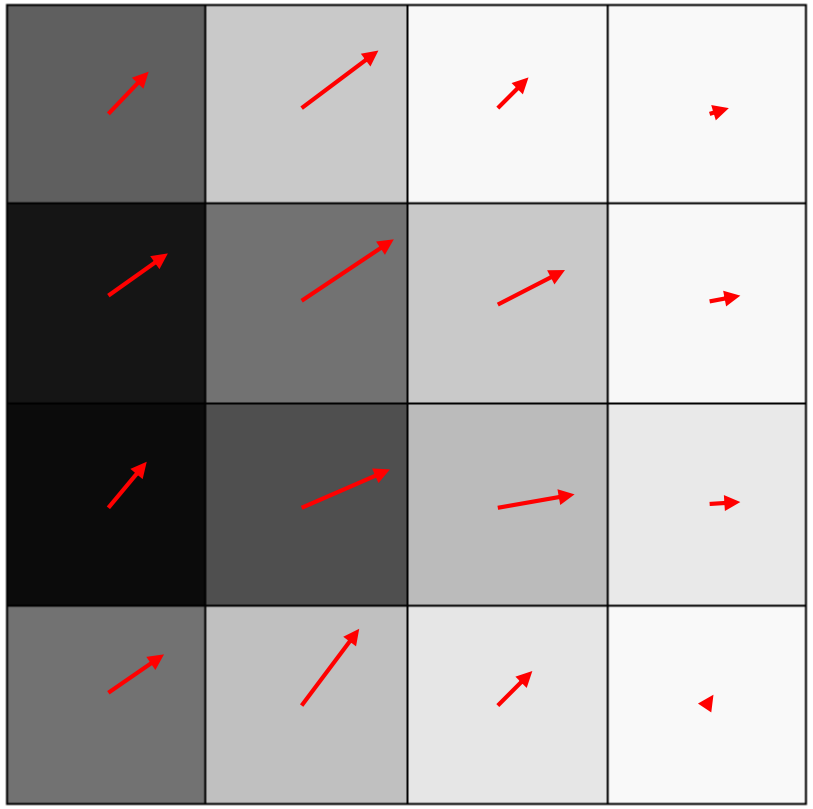


$$\text{magnitude(x,y)}=\sqrt{\left(L(x+1,y)-L(x-1,y)\right)^2+\left(L(x,y+1)-L(x,y-1)\right)^2}$$

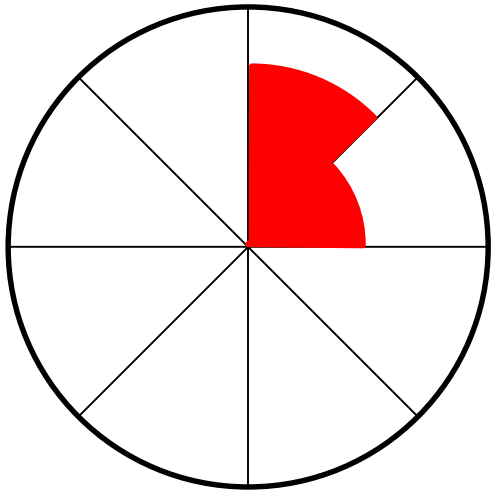
$$\theta(\text{x,y})=\text{a}tan\left(\left(\frac{\text{L(x,y+1)}-\text{L(x,y-1)}}{\text{L(x+1,y)}-\text{L(x-1,y)}}\right)\right)$$

by Yao Lu

Numeric Example



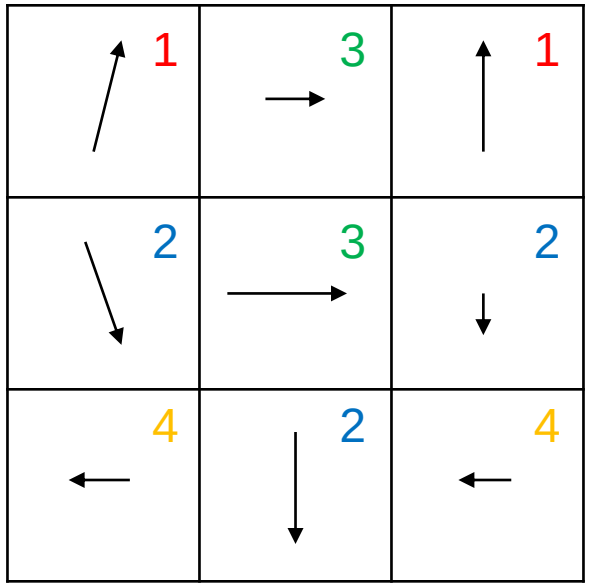
Orientations in each of the 16 pixels of the cell



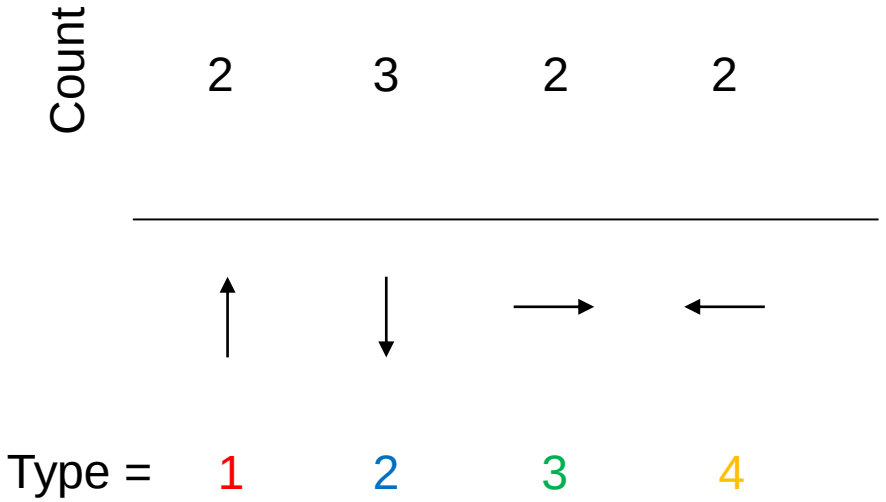
The orientations all ended up in two bins: 11 in one bin, 5 in the other. (rough count)

5 11 0 0 0 0 0 0

Scale Invariant Feature Transform



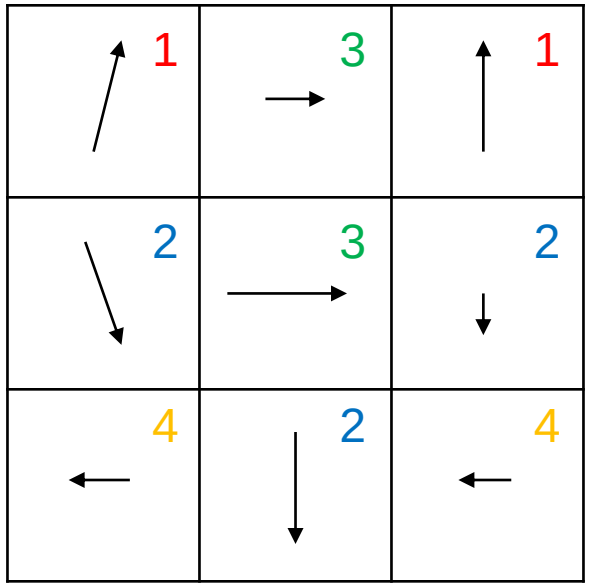
Uniform weight (ignore magnitude)



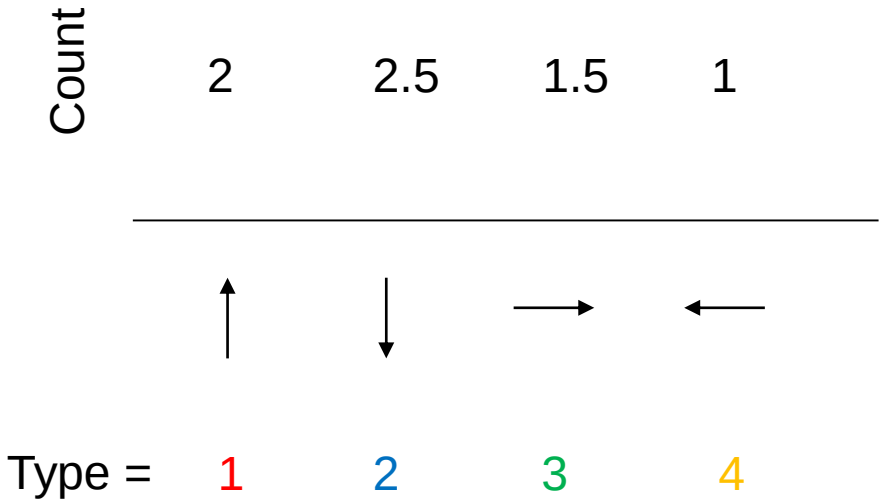
Gradients

Histogram of gradients

Scale Invariant Feature Transform



Weight contribution by magnitude (e.g. long = 1, short = 0.5)



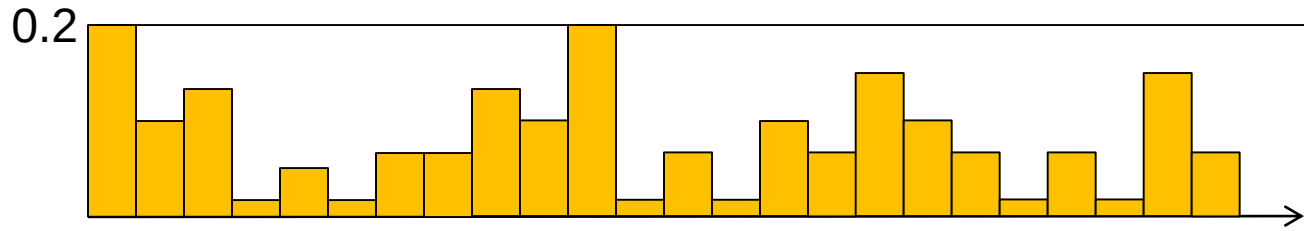
Gradients

Histogram of gradients

Scale Invariant Feature Transform

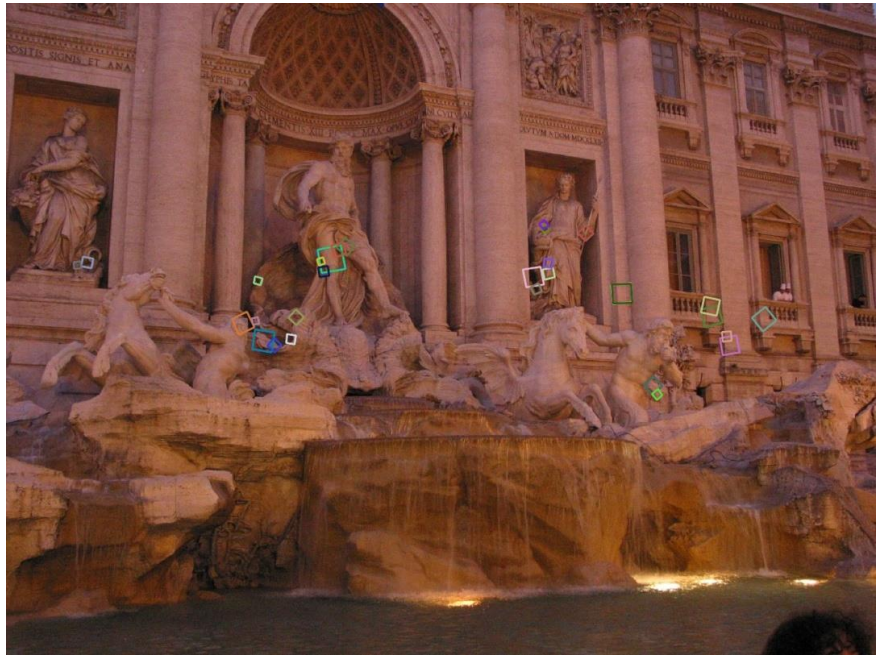
- Adding robustness:
  - The descriptor is made of gradients (differences between pixels), so it's already invariant to changes in brightness (e.g., adding 10 to all image pixels yields the exact same descriptor)
  - A higher-contrast photo will increase the magnitude of the gradients linearly. So, the correct for contrast changes, notmalize the vector (scale to length 1.0)
  - Very large image gradients are usually from unreliable 3D illumination effects (glare, etc.). So to reduce their effect, clamp all values in the vector to be ≤ 0.2 (an experimentally tunes value). Then normalize the vector again
  - After normalizing, we have:

$$\sum_i d_i = 1 \quad \text{such that:} \quad d_i < 0.2$$

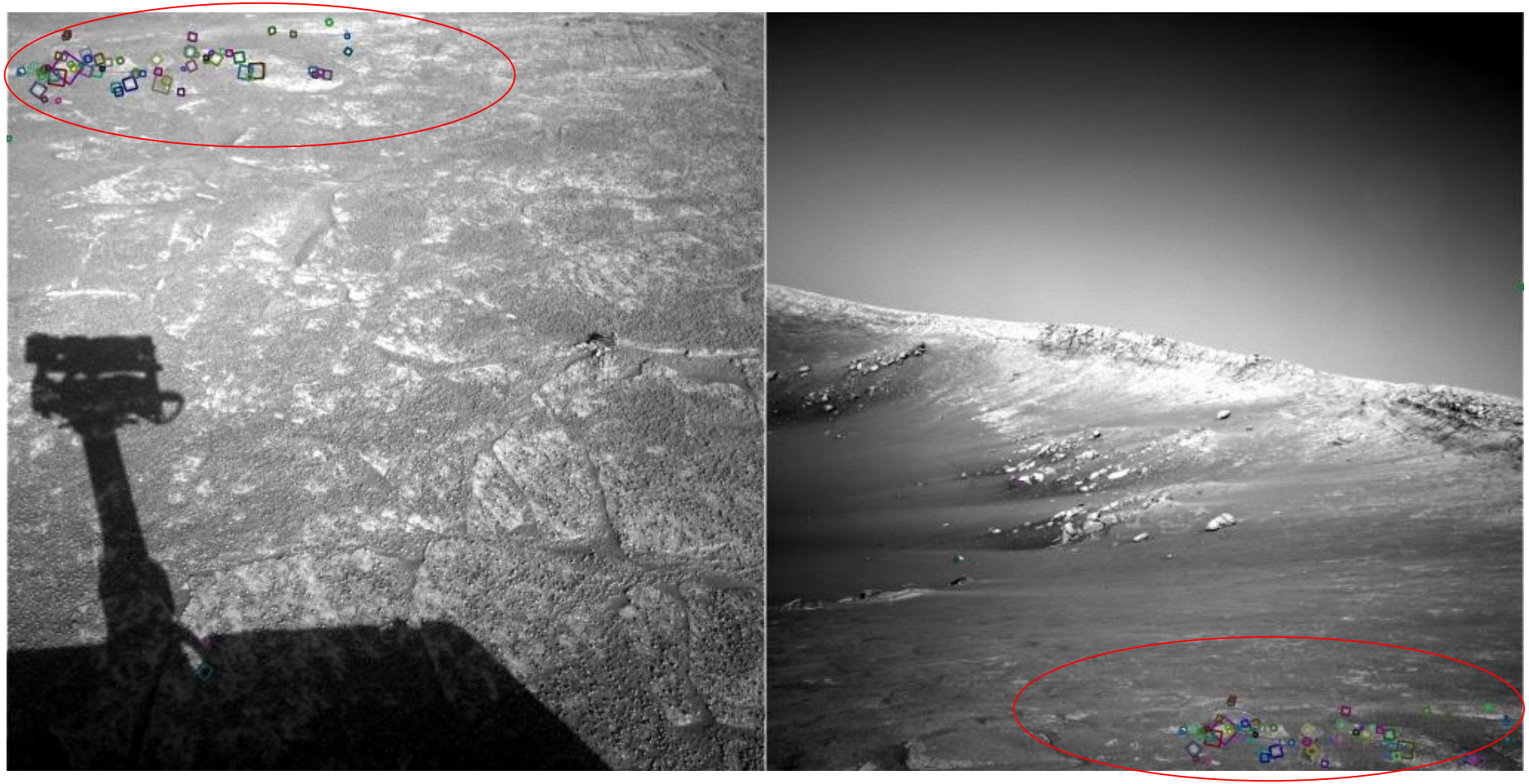


Properties of SIFT

- Extraordinarily robust matching technique
  - Can handle changes in viewpoint
    - Up to about 30 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
  - Various code available
    - http://www.cs.ubc.ca/~lowe/keypoints/



Example



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

Example: Object Recognition

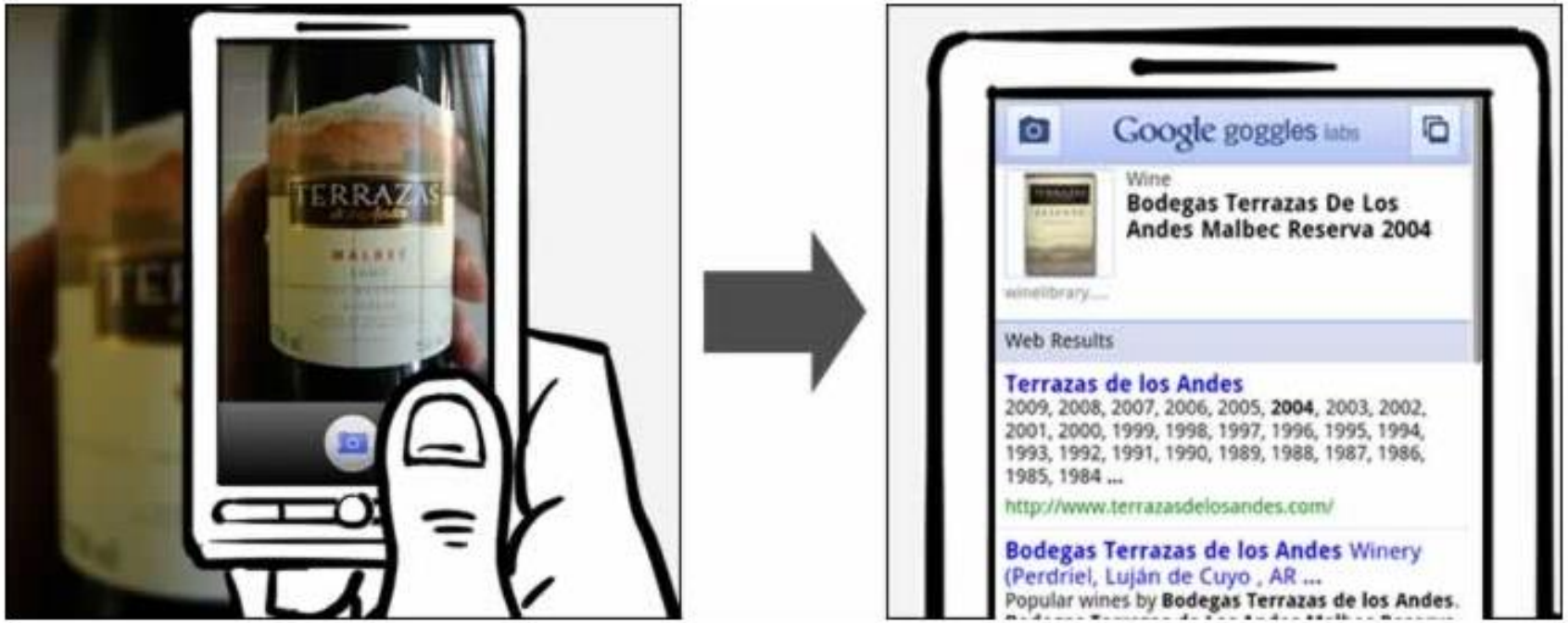
SIFT is extremely powerful for object instance recognition, especially for well-textured objects

Lowe, IJCV04

Example: Google Goggle

Google Goggles in Action

Click the icons below to see the different ways Google Goggles can be used.

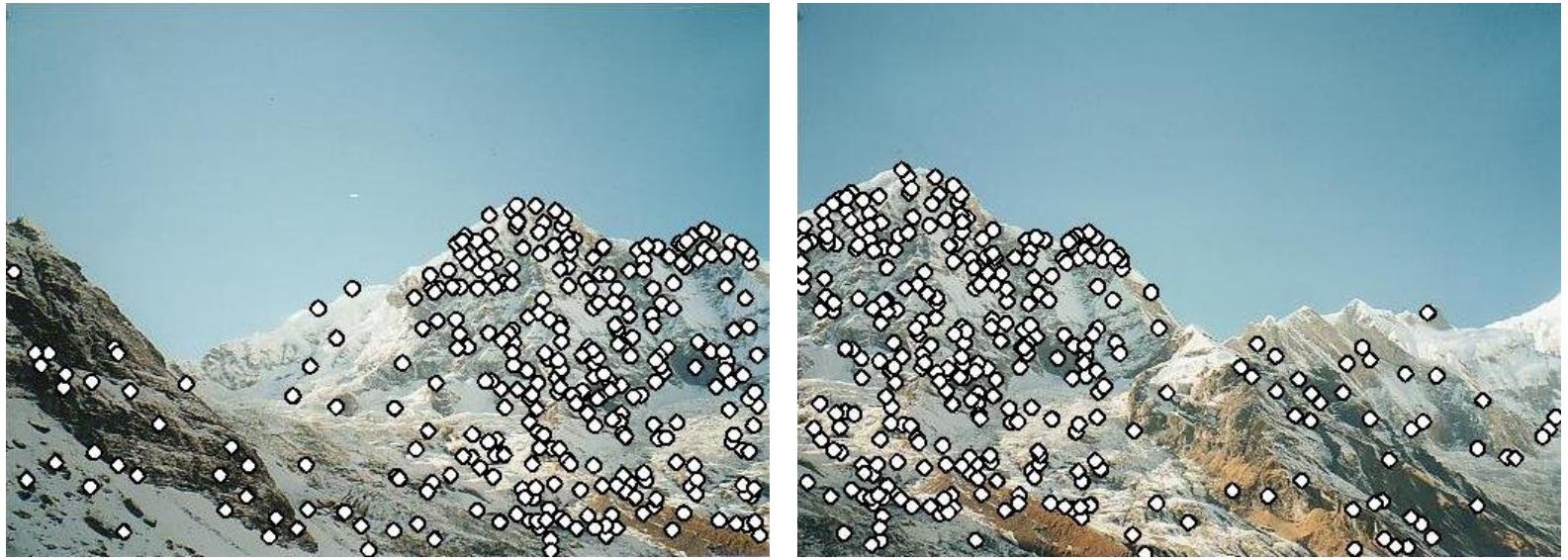


Panorama?

- We need to match (align) images

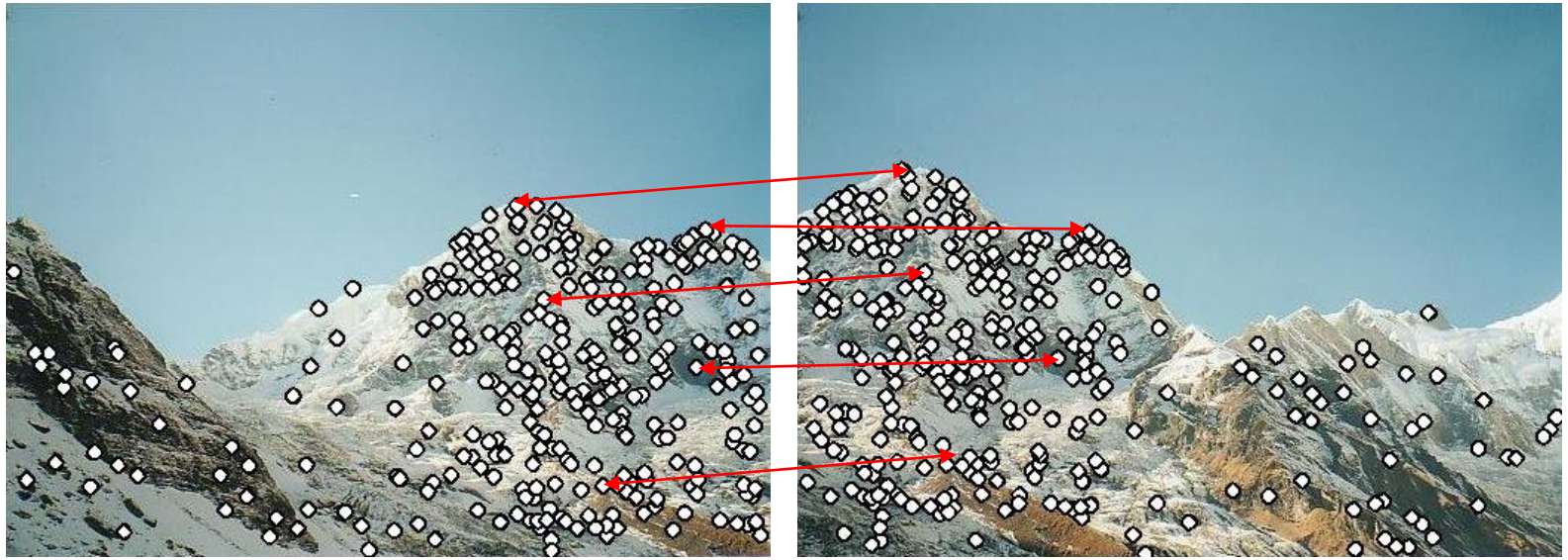
Matching with Features

- Detect feature points in both images



Matching with Features

- Detect feature points in both images
- Find corresponding pairs

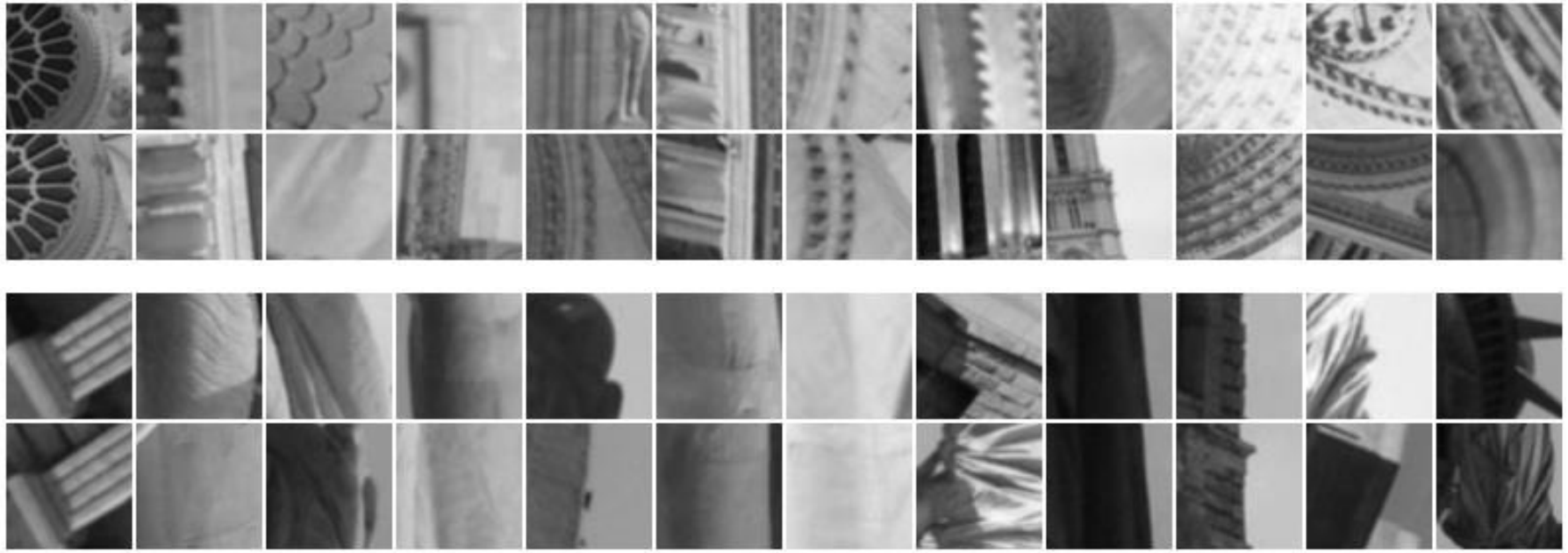


Matching with Features

- Detect feature points in both images
- Find corresponding pairs
- Use these matching pairs to align images - the required mapping is called a **homography**.

When does the SIFT descriptor fail?

- Patches SIFT thought were the same but aren't:



Other kinds of descriptors

- SIFT is predominately used for matching

- SIFT descriptors are computed at sparse, scale-invariant keypoints and are rotated to align orientation.

- There are descriptors for other purposes
  - Describing shapes
  - Describing textures

HISTOGRAM OF ORIENTED GRADIENTS

N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection", Proc. IEEE Conf. Computer Vision and Pattern Recognition, vol. 1, pp. 886-893, 2005.

Dalal and Triggs introduced it in 2005.
Their work has been cited over 13,000 times!
Used for object classification in 2006

Histogram of Oriented Gradients (HoG)

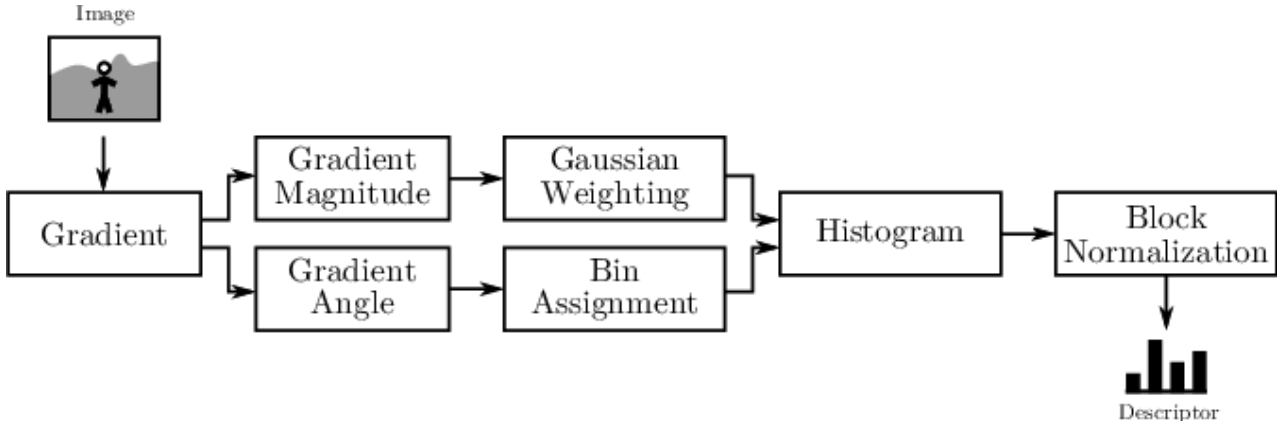
- Find robust feature set that allows object form to be discriminated.

- Challenges
  - Wide range of pose and large variations in appearances
  - Cluttered backgrounds under different illumination
  - “Speed” for mobile vision

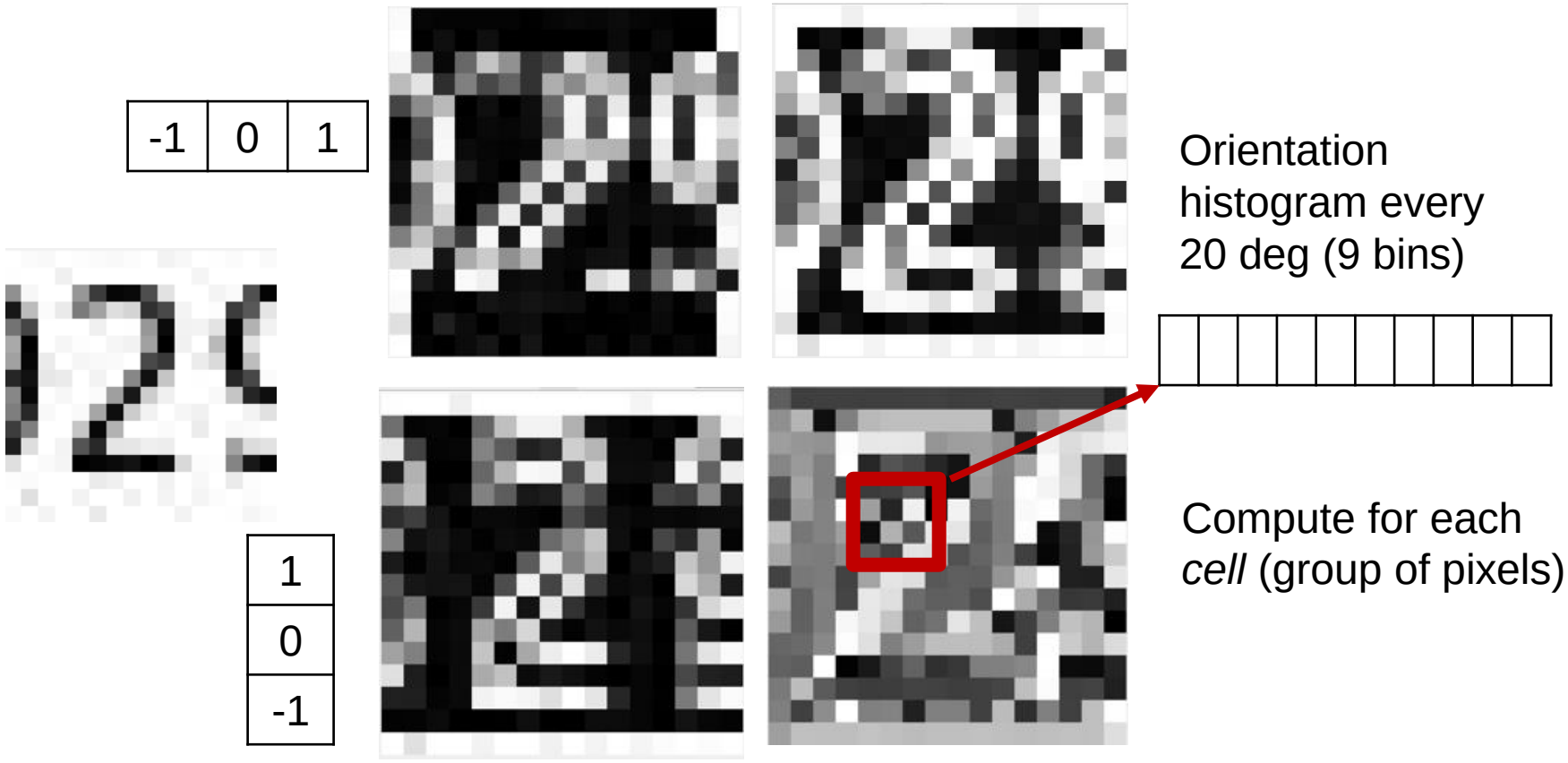
- Local object appearance and shape can often be characterized rather well by the distribution of local intensity gradients or edge directions

Histogram of Oriented Gradients (HoG)

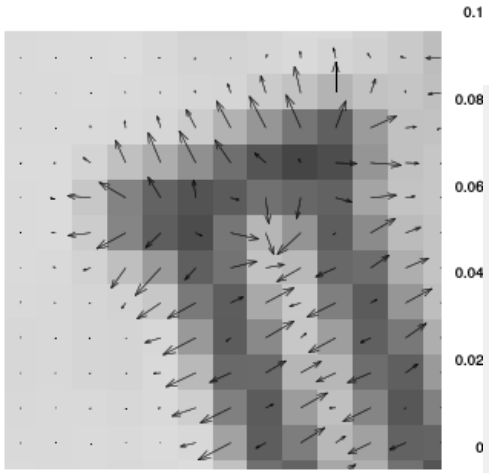
- Steps:
  1. Gradient computation
  2. Orientation binning
  3. Descriptor blocks
  4. Block normalization
- HOG descriptors may be used for object recognition by providing them as features to a machine learning algorithm.

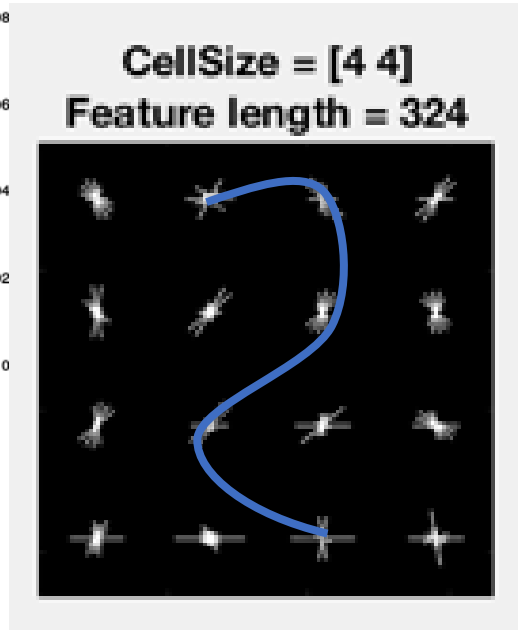


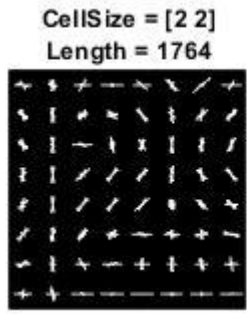
1. Calculate edge gradient magnitude and direction

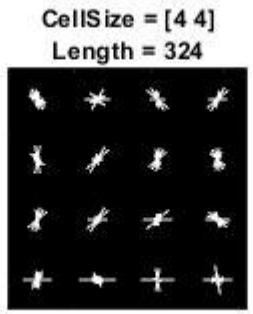


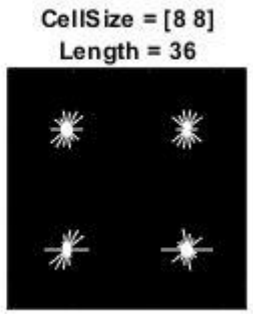
1b. Edge direction histogram visualization



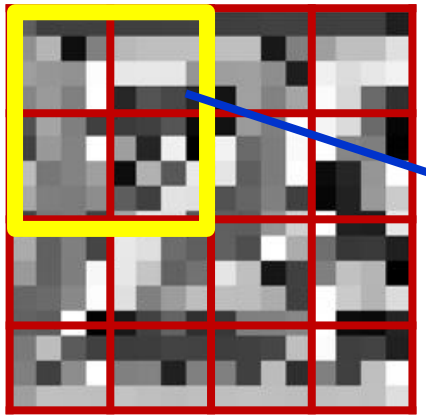








2. Repeat for each cell in block



This yellow block has 2x2=4 non-overlapping cells, each 16 pixels.

Orientation histogram for yellow block:

9 x 4 = 36 features

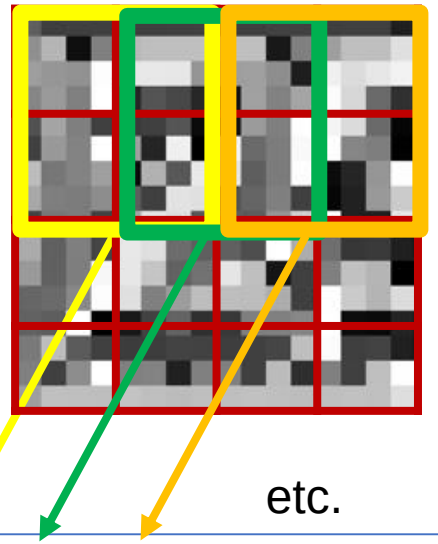
Then normalize feature vector,

$$n = \frac{v}{\sqrt{\|v\|^2 + \epsilon}}$$

For example, this gives unit length.

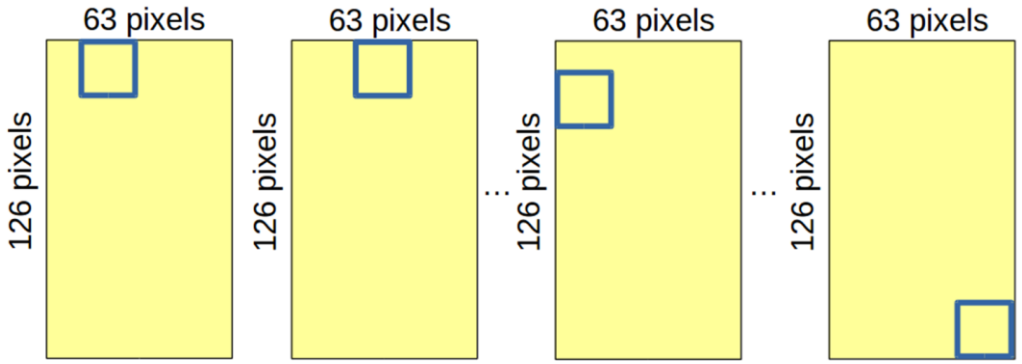
- This normalization is done in order to remove the effect of local lights differences

3. Repeat for each block of cells in the window



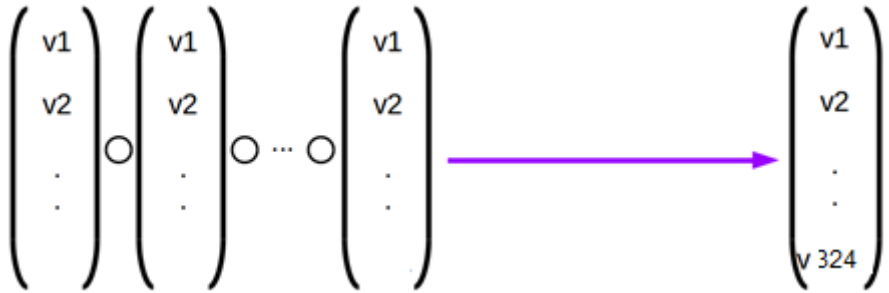
etc.

Window has *blocks* of *cells* made of *pixels*.



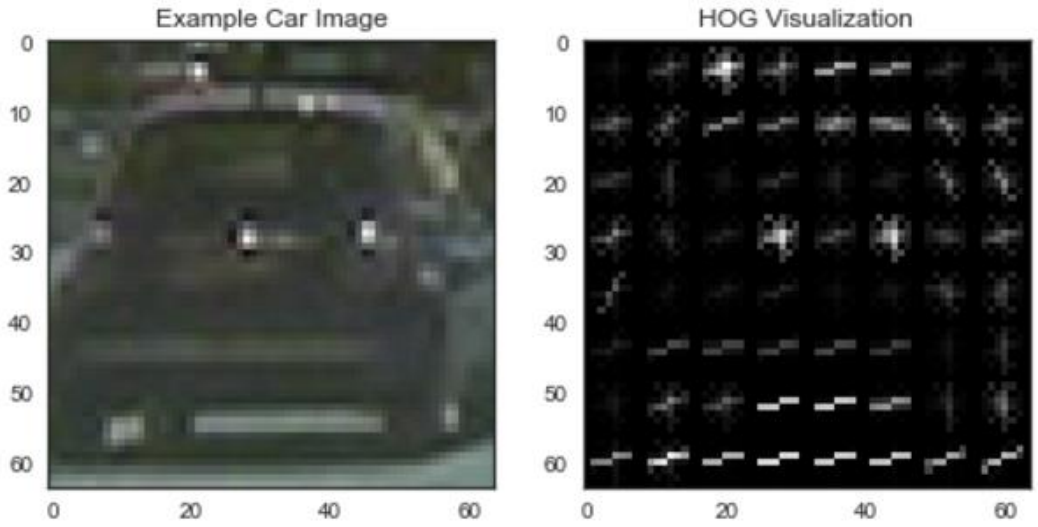
36/block x 9 overlapping blocks = 324 features

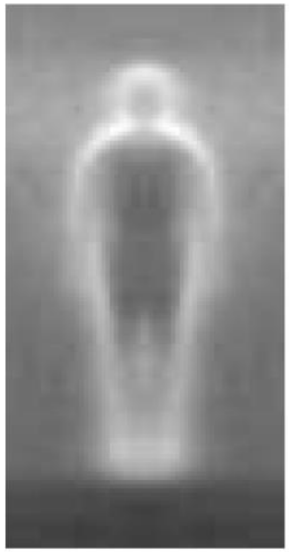
Each block is normalized independently

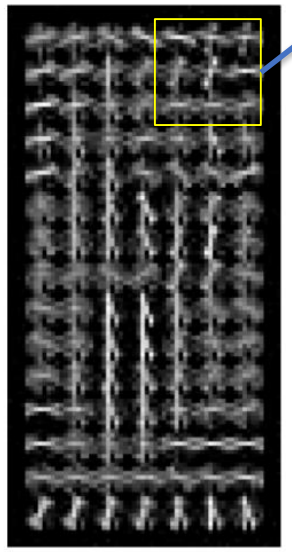


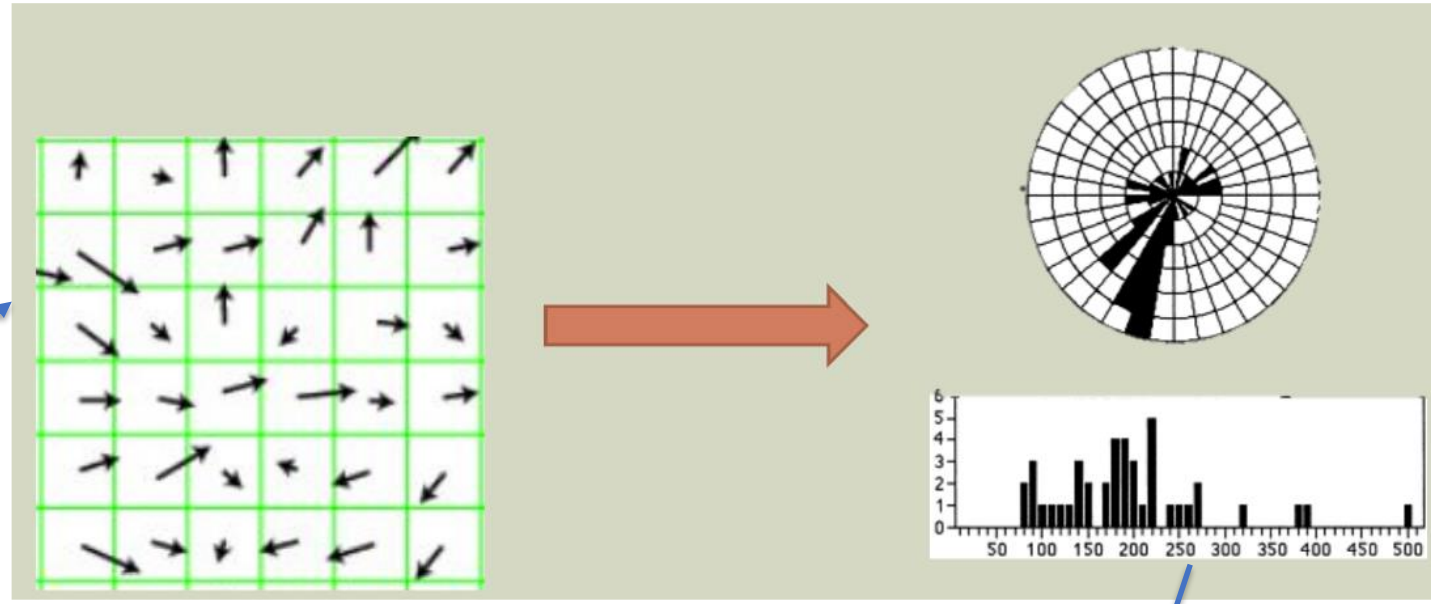
https://www.mathworks.com/help/vision/examples/digit-classification-using-hog-features.html

Histogram of Oriented Gradients (HoG)





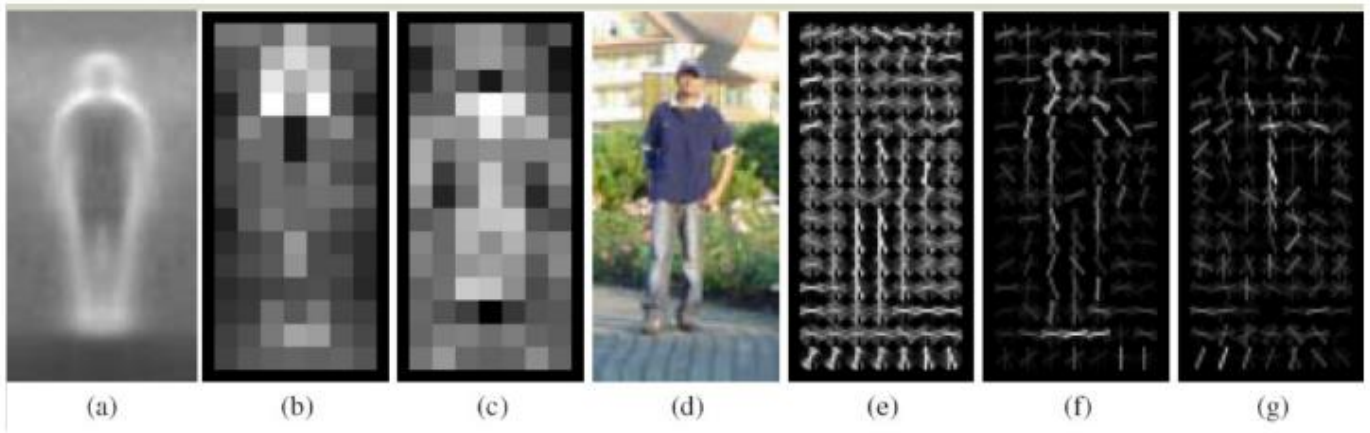




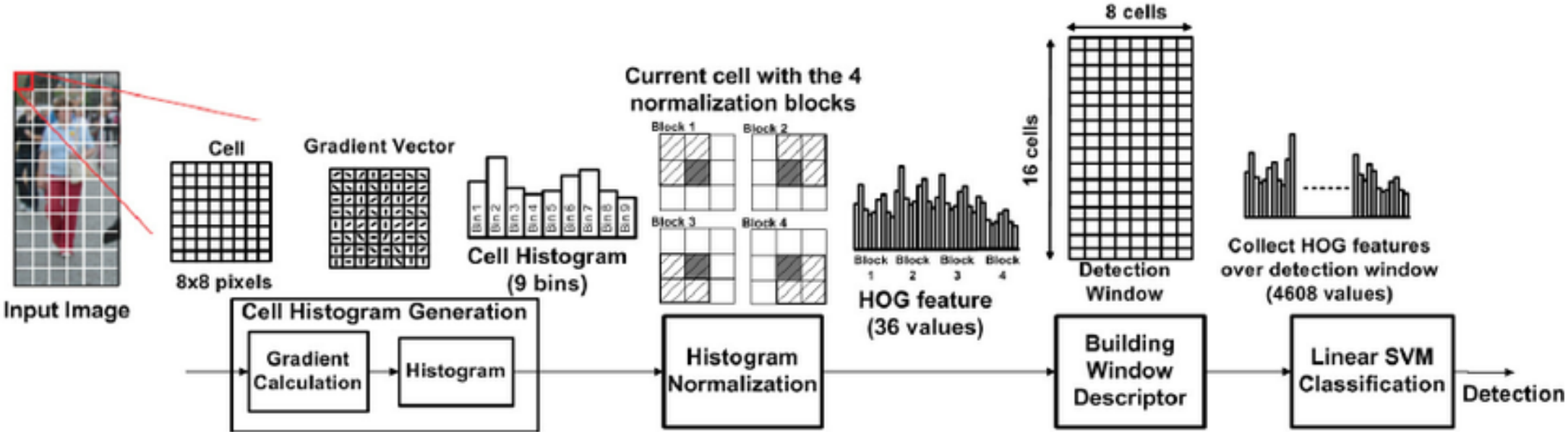
Feature Vector

Visualizing HoG

a. Average gradient over positive examples
b. Maximum positive weight in each block
c. Maximum negative weight in each block
d. A test image
e. It's R-HOG descriptor
f. R-HOG descriptor weighted by positive weights
g. R-HOG descriptor weighted by negative weights



HoG for detection



HoG for detection

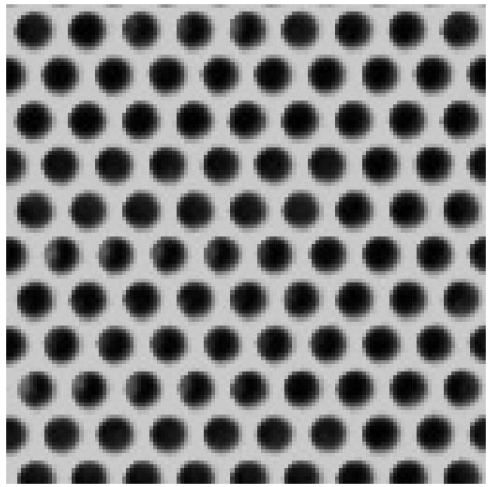
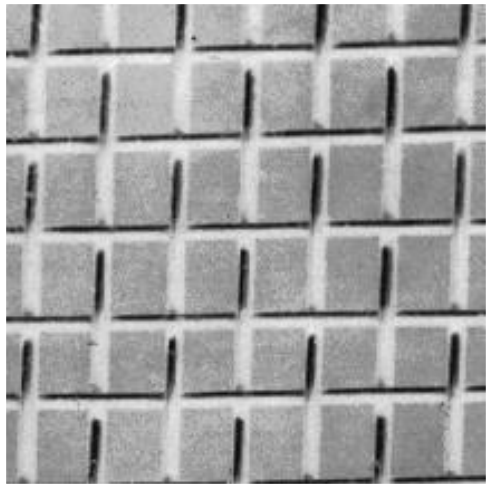
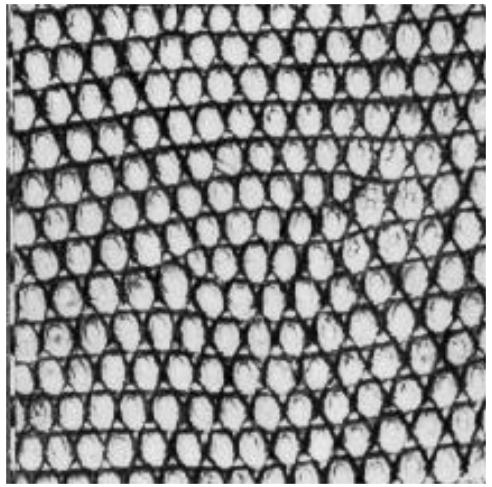
Difference between HoG and SIFT

- HoG is usually used to describe entire images. SIFT is used for key point matching
- SIFT histrograms are oriented towards the dominant gradient. HoG is not.
- HoG gradients are normalized using neighborhood bins.
- SIFT descriptors use varying scales to compute multiple descriptors.

LOCAL BINARY PATTERNS

Texture

- The texture features of a patch can be considered a descriptor.
- E.g. the LBP histogram is a texture descriptor for a patch.

Pixel Neighborhood-based Feature

- The most important for texture analysis is to describe the spatial behavior of intensity values in any given neighborhood.
- Different methodologies have been proposed.
- Local binary pattern (LBP) is one of the most-widely used approach – mainly for face recognition.
- LBP is used for texture analysis too.

Local Binary Pattern (LBP)

- **Local Binary Pattern** (LBP) is a simple yet very efficient texture operator

- Labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number.

- Due to its discriminative power and computational simplicity, LBP texture operator has become a popular approach in various applications.

LBP in the spatial domain

- The original LBP operator (Ojala et al. 1996) forms labels for the image pixels by thresholding the 3 x 3 neighborhood of each pixel with the center value and considering the result as a binary number.

- This operator used jointly with a simple local contrast measure provided very good performance in unsupervised texture segmentation (Ojala and Pietikäinen 1999).

- After this, many related approaches have been developed for texture and color texture segmentation.

LBP Extensions

- The LBP operator was extended to use neighborhoods of different sizes (Ojala et al. 2002).

- Using a circular neighborhood and bilinearly interpolating values at non-integer pixel coordinates allow any radius and number of pixels in the neighborhood.

- The gray scale variance of the local neighborhood can be used as the complementary contrast measure. In the following, the notation (P, R) will be used for pixel neighborhoods which means P sampling points on a circle of radius of R.

Local Binary Pattern (LBP)

- For each PIXEL of an image, a BINARY CODE is produced
  - to make a new matrix with the new value (binary to decimal value).

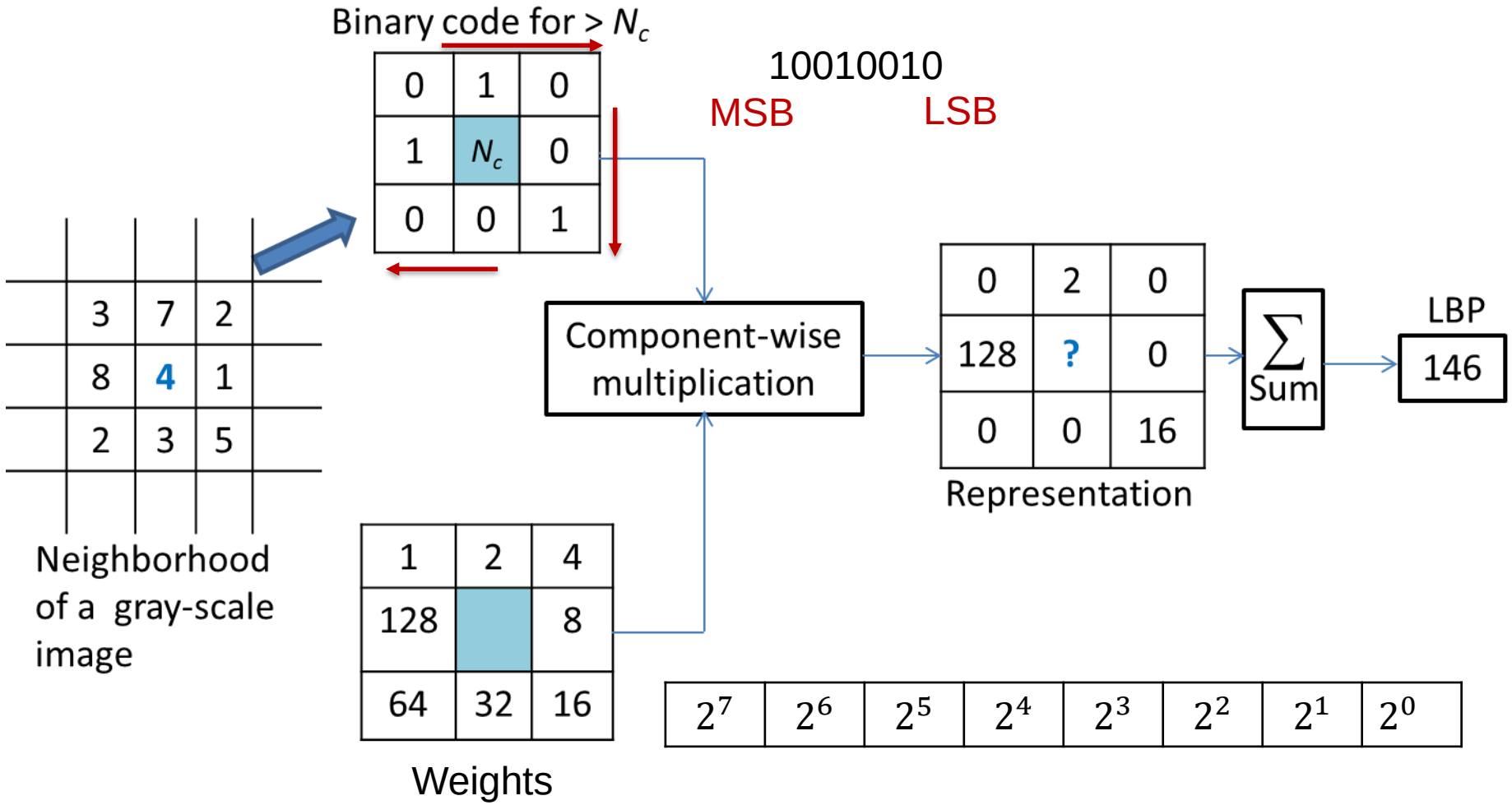
$$LBP_{p,r}(N_c) = \sum_{p=0}^{P-1} g\left(N_p - N_c\right)2^p$$

- *where,*
- neighborhood pixels (N_p) in each block →
- is thresholdedby its center pixel value (N_c)
- p→ sampling points (e.g., $p = 0, 1, \ldots, 7$ for a 3x3 cell, where P = 8)
- r → radius (for 3x3 cell, it is 1).

Binary threshold function $g(x)$ is,

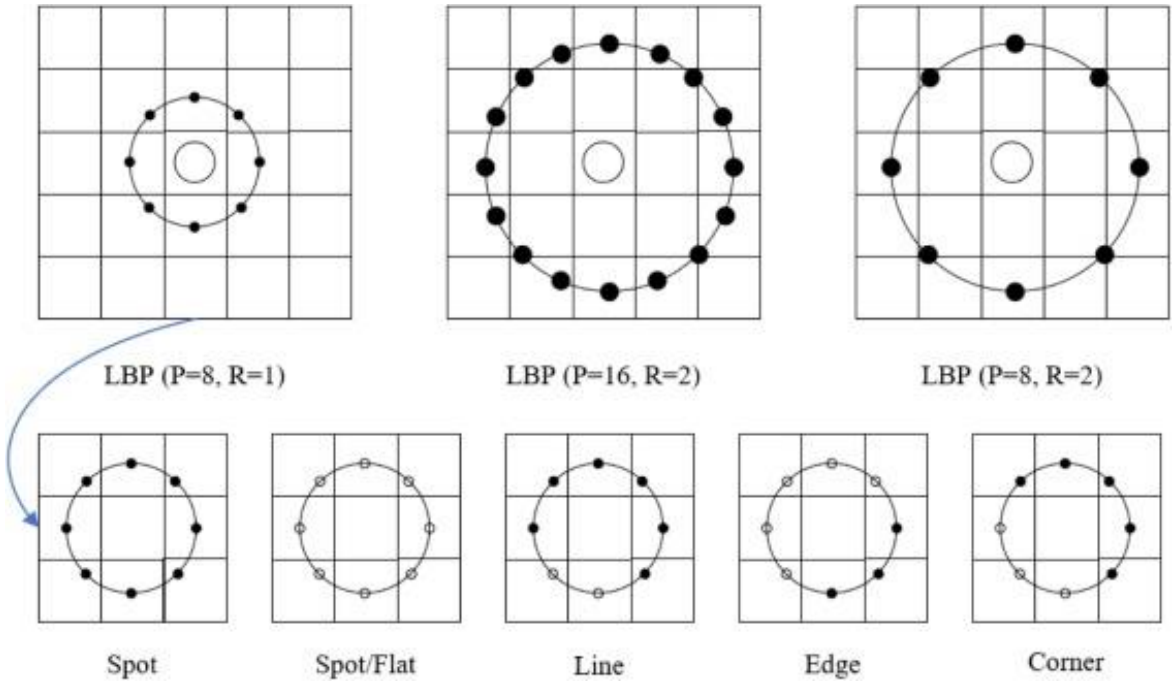
$$g(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$$

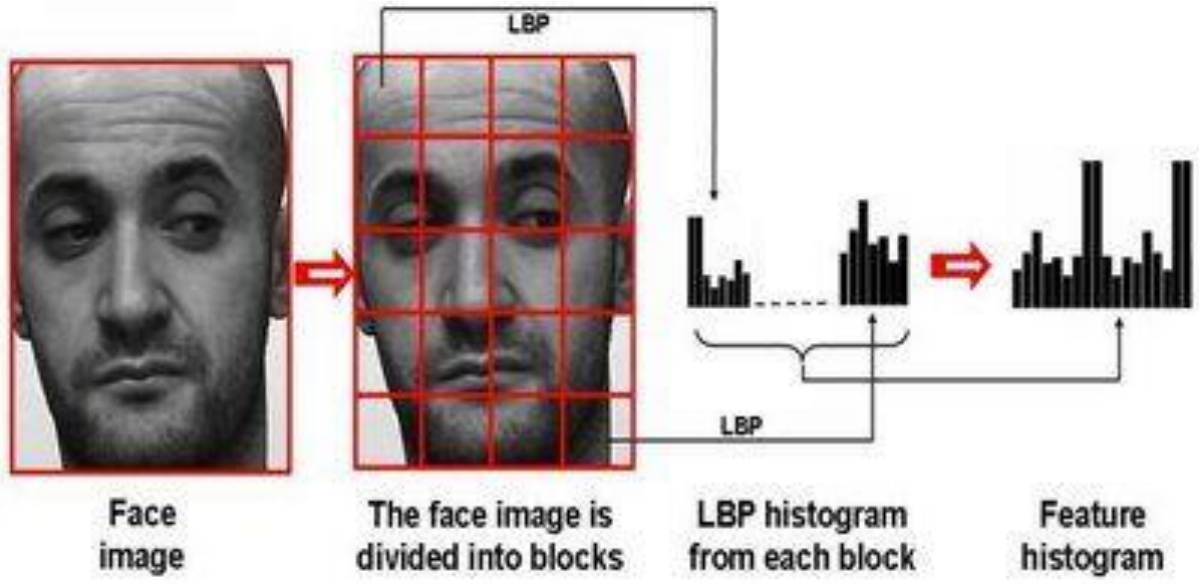
Computation of Local Binary Pattern



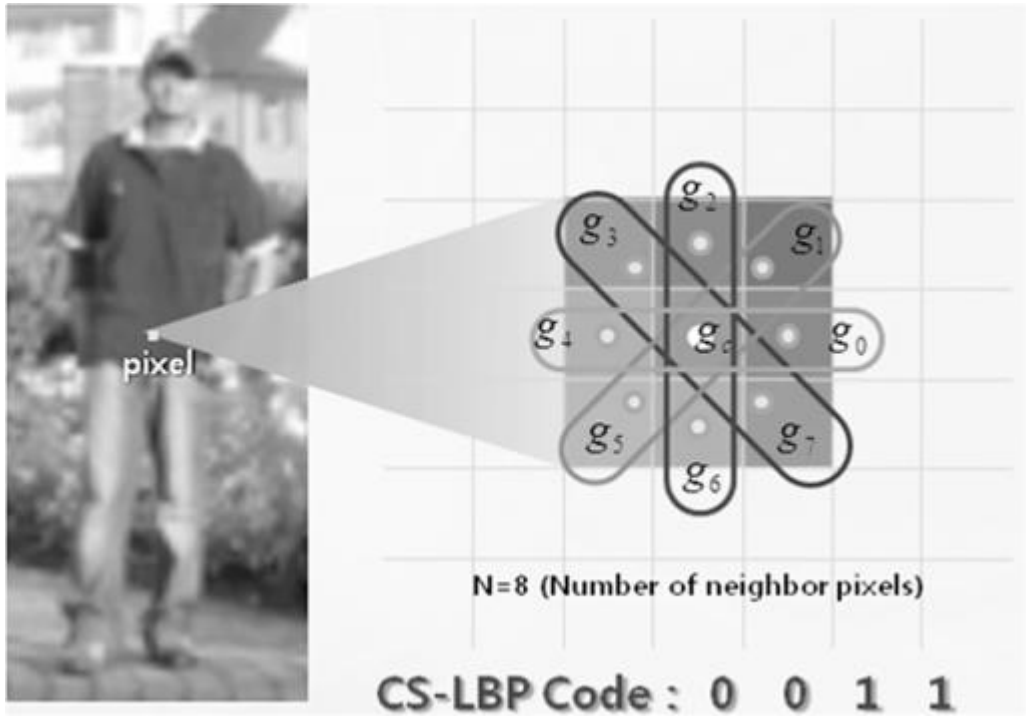
Example of how the *LBP operator* works

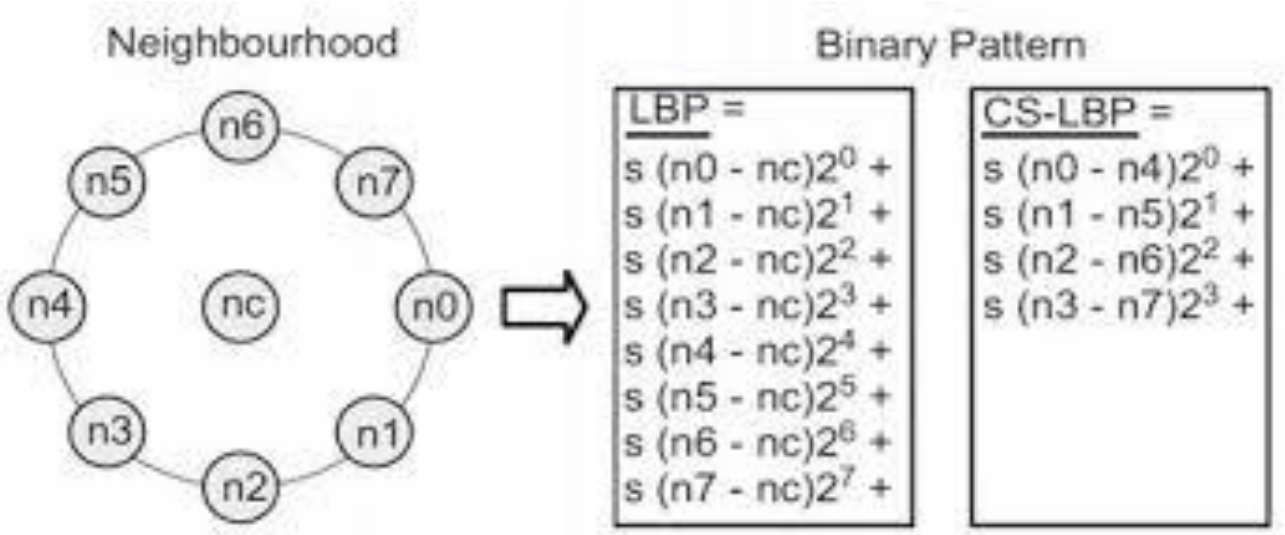
LBP Overview





Center-Symmetric LBP





What we have learned today?

- Keypoint detection: repeatable and distinctive
  - Corners, blobs, stable regions
  - Laplacian of Gaussian, automatic scale selection
- Descriptors: robust and selective
  - Histograms for robustness to small shifts and translations (SIFT descriptor)
- HoG
  - Another image descriptor
  - Finds application in object detection
  - We will revisit this!
- Local Binary Patterns for texture analysis