Master Mathematical Analysis and Applications
Course M1 - S2

Computer vision

Key-points extraction, description, and matching

Week 6-8

Mohammed Hachama
hachamam@gmail.com
http://hachama.github.io/home/
University of Khemis Miliana

-2020-

Plan

1. Keypoints extraction, description and matching

Keypoints

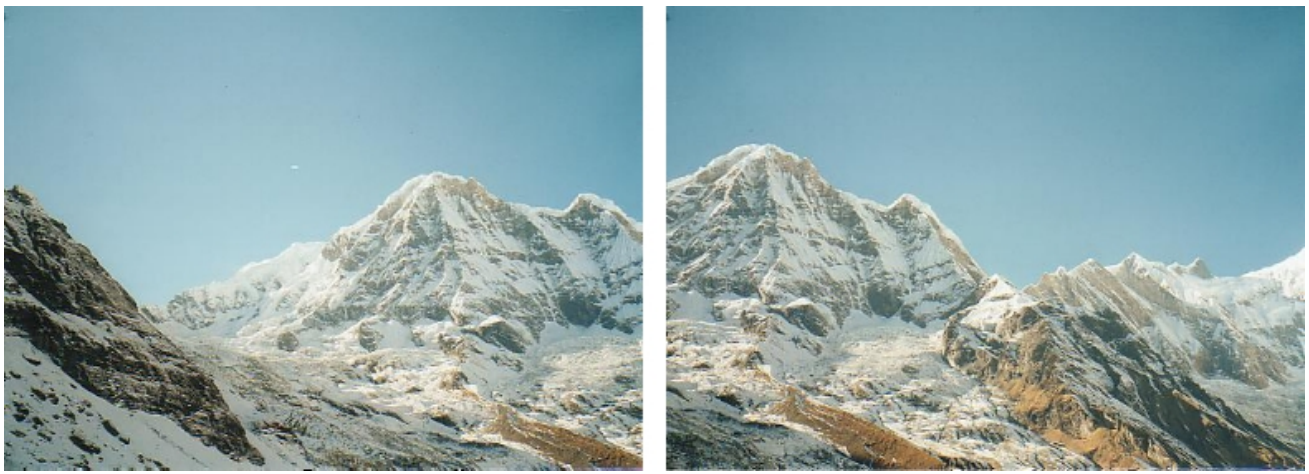
What is a good Keypoint ?

What is a good Keypoint ?

- Repeatability and stability
  - The same keypoint can be found in several images despite geometric and photometric transformations. These points usually don't keep moving around in the image. This helps tracking.
- Saliency (Uniquely identifiable)
  - Each keypoint is distinctive
- Compactness and efficiency
  - Many fewer keypoints than image pixels
- Locality
  - A keypoint occupies a relatively small area of the image ; robust to clutter and occlusion

Applications

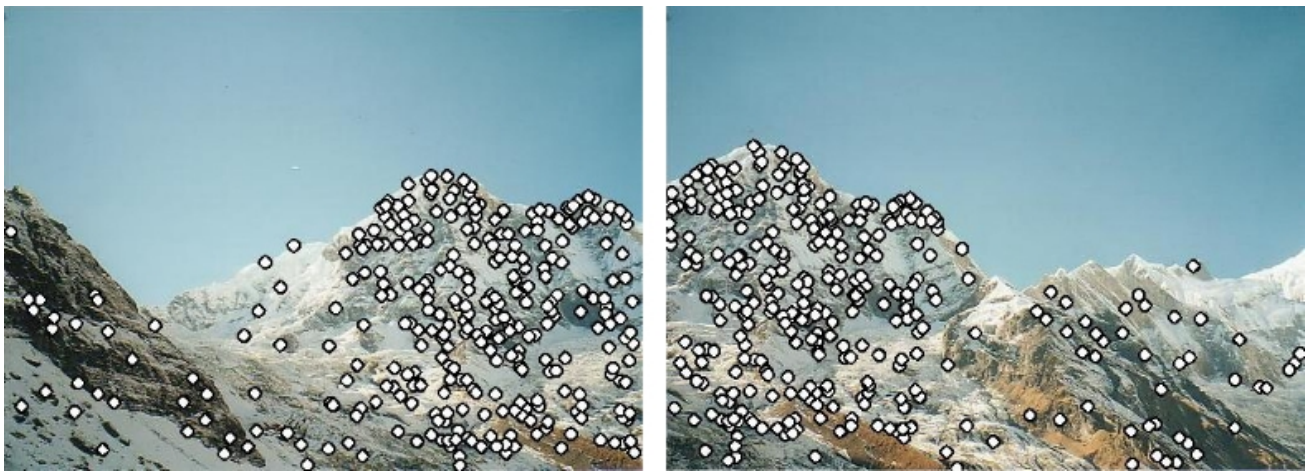
Panorama stitching



- Step 1 : extract keypoints
- Step 2 : match keypoint features
- Step 3 : align images

Applications

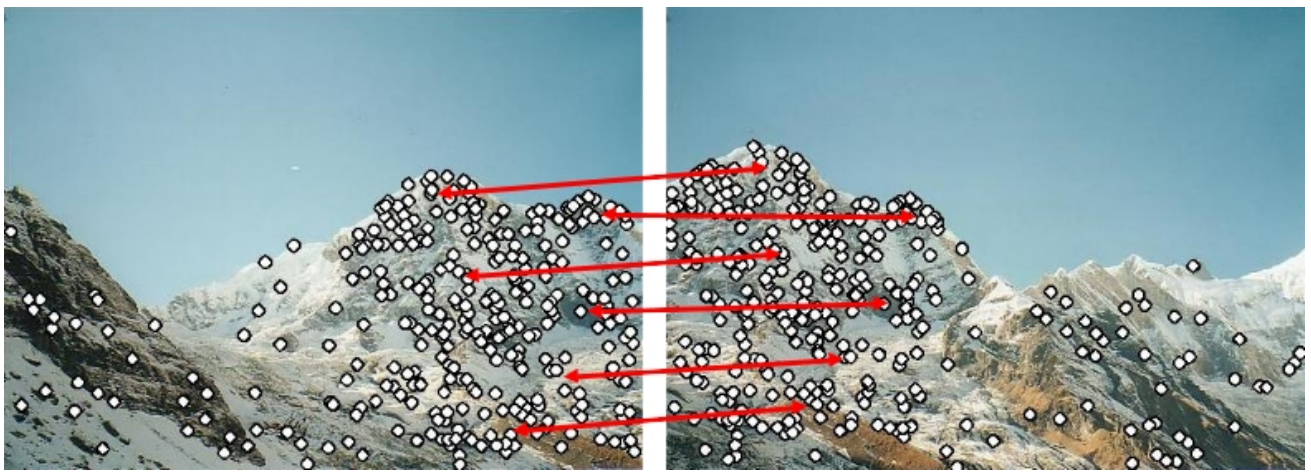
Panorama stitching



- Step 1 : extract keypoints
- Step 2 : match keypoint features
- Step 3 : align images

Applications

Panorama stitching



- Step 1 : extract keypoints
- Step 2 : match keypoint features
- Step 3 : align images

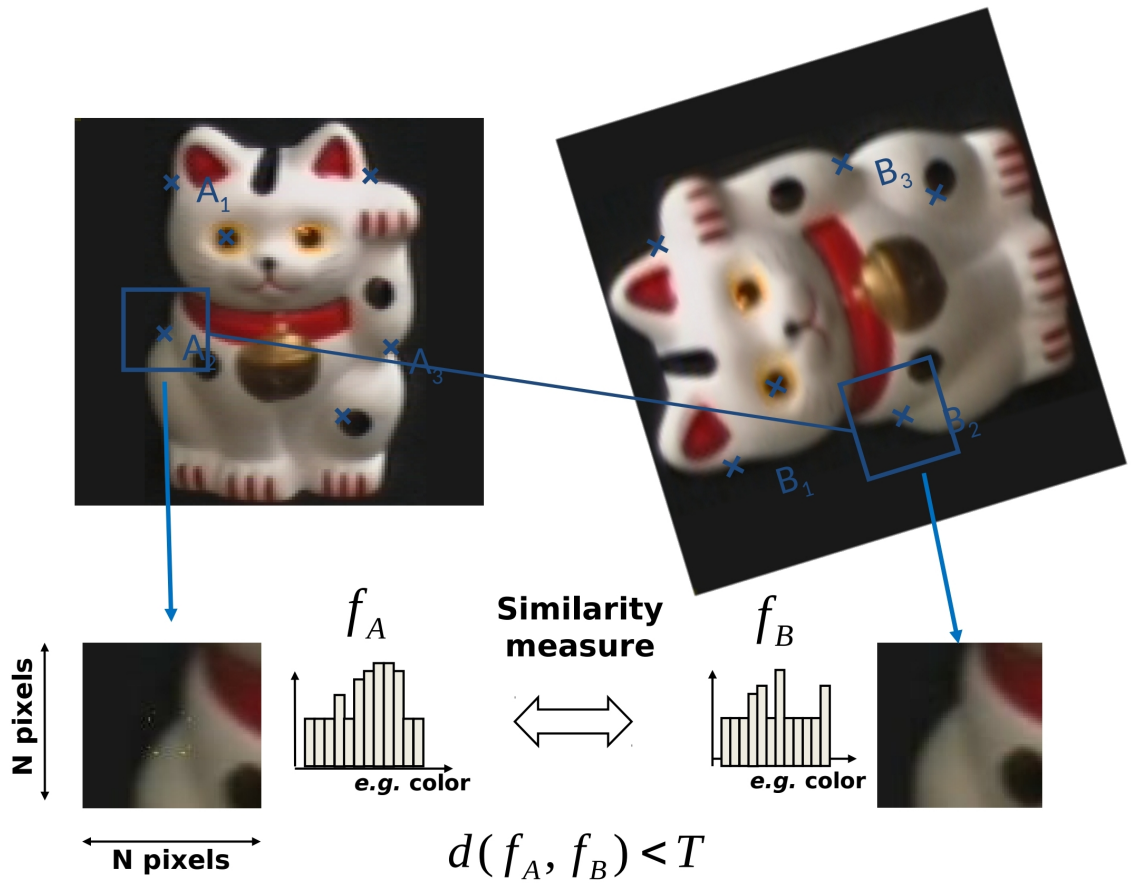
Applications

Panorama stitching

- Step 1 : extract keypoints
- Step 2 : match keypoint features
- Step 3 : align images

Applications

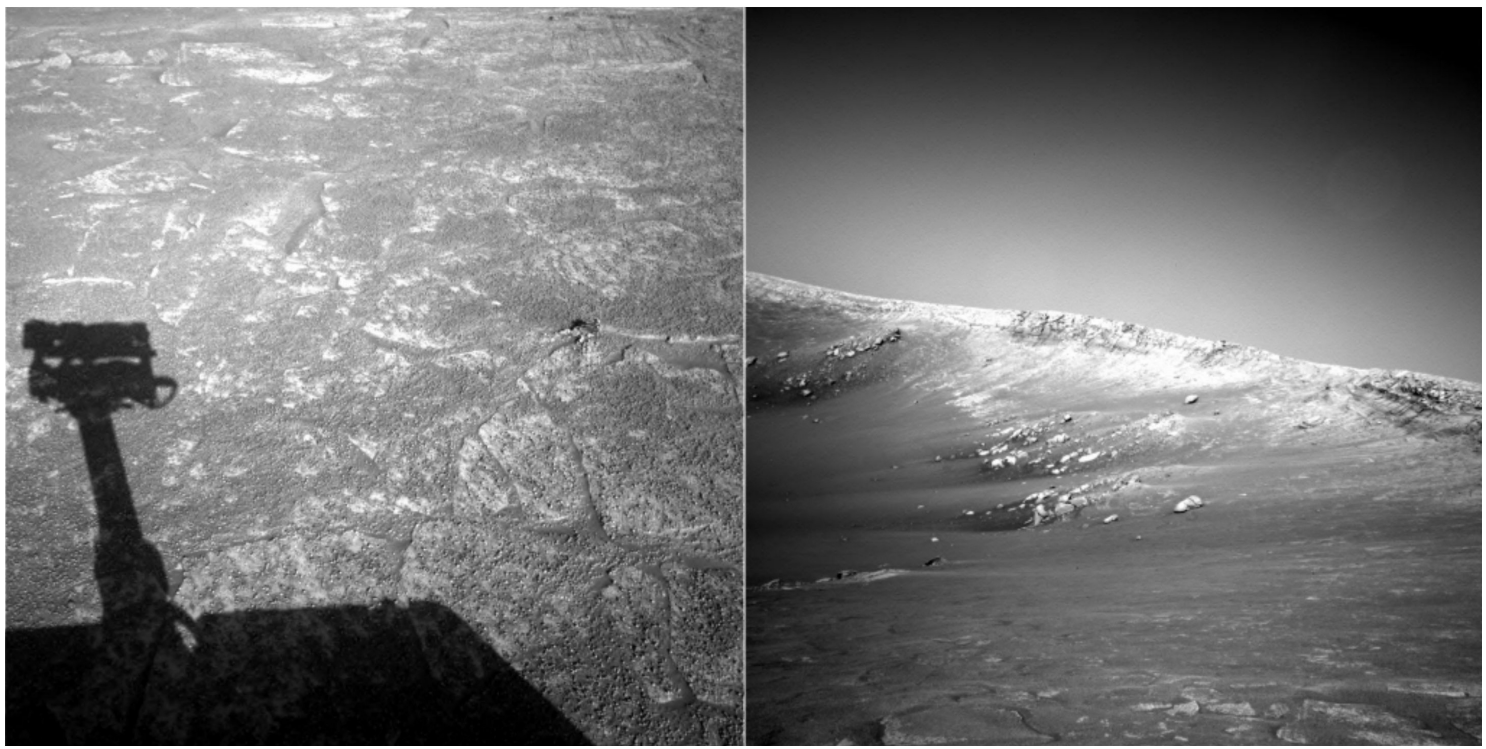
Matching



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

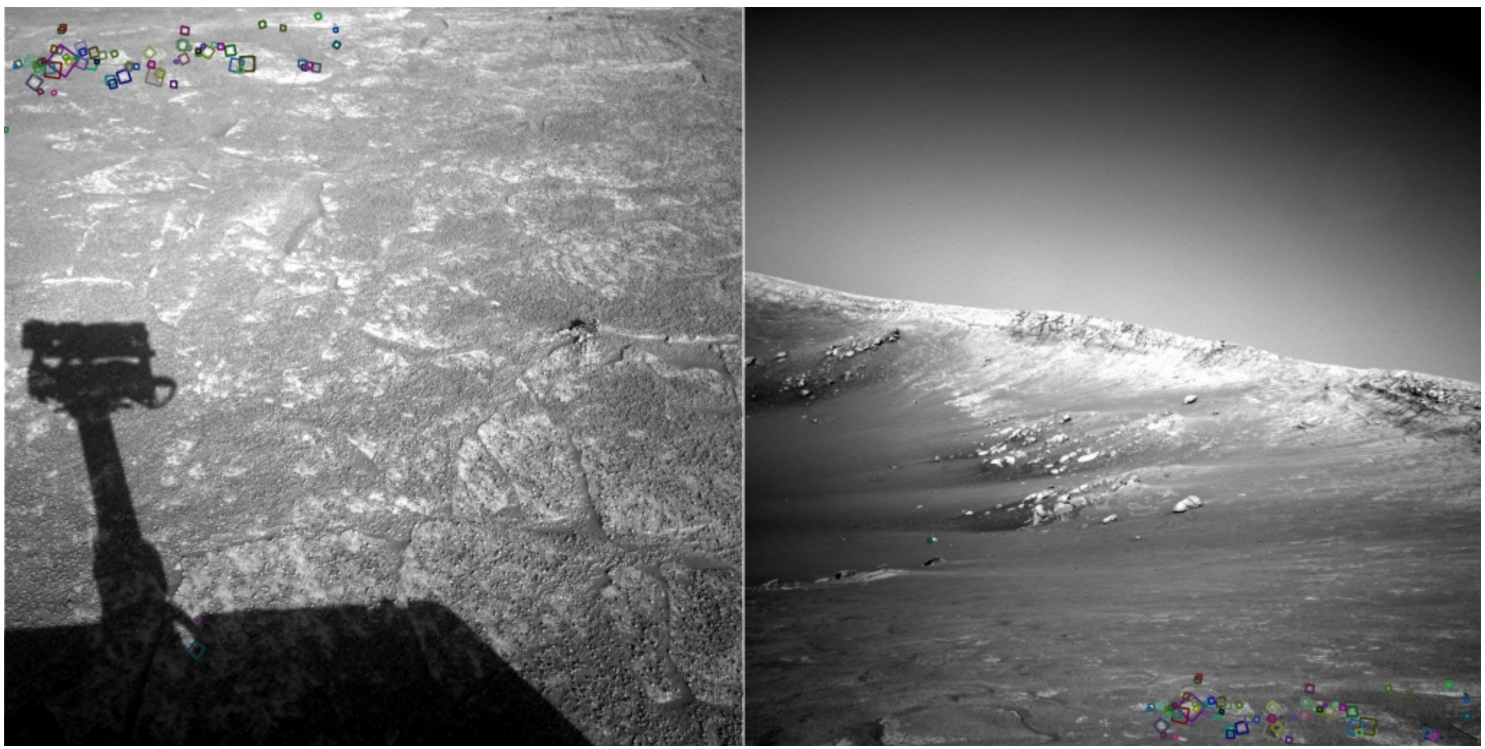
Applications

Matching



Applications

Matching

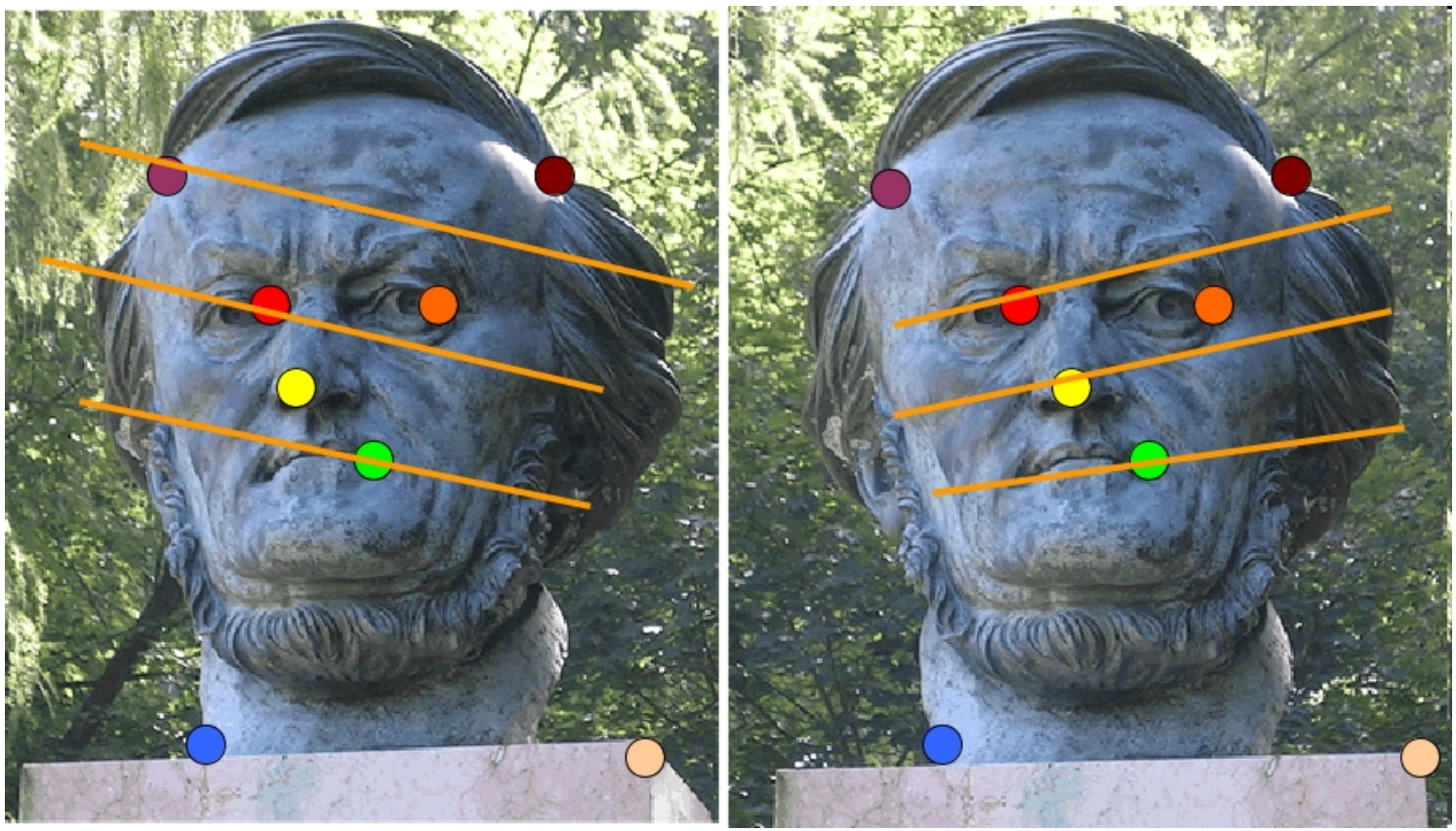


Applications

3D reconstruction : Structure from motion

Applications

Calibration : estimating “fundamental matrix” that corresponds two views



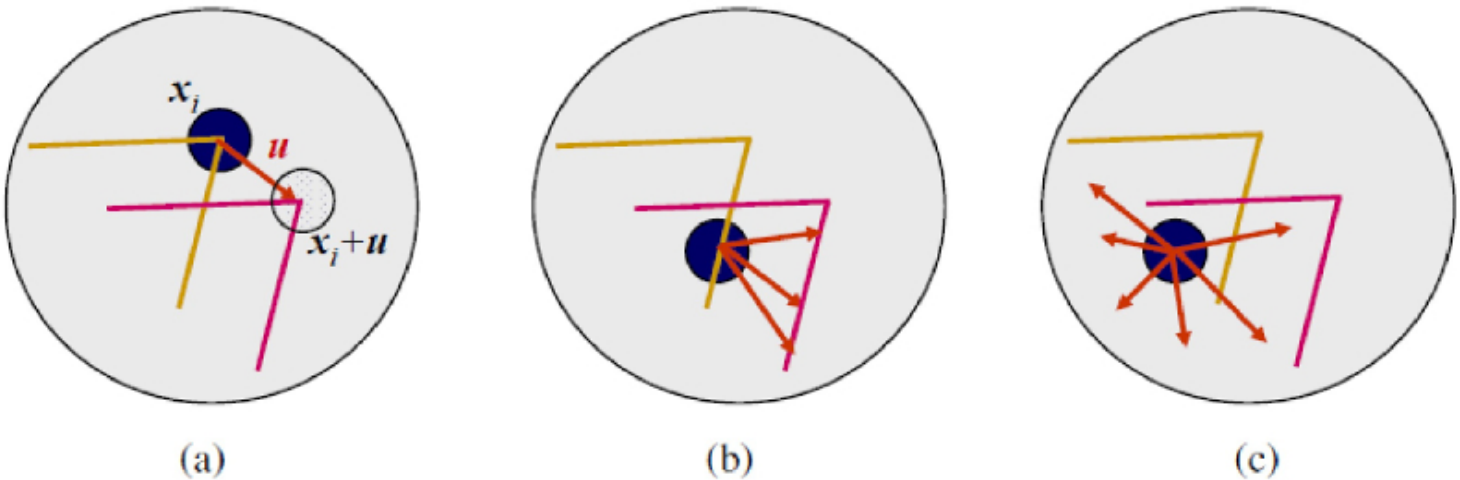
Applications

- Image alignment
- Motion tracking
- Robot navigation
- Indexing and database retrieval
- Object recognition
- 3D reconstruction

Corner Detection

Keypoints : Corners

- Aperture Problem



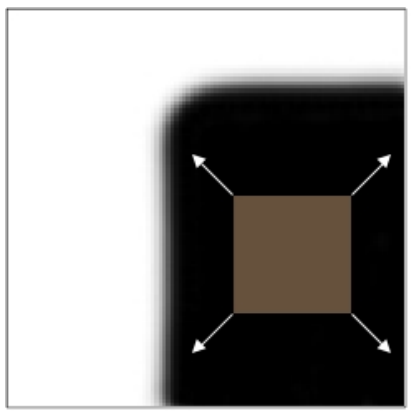
Corner Detection

- Based on boundary extraction
  - First step edge detection
  - Curvature analysis of edges
- Based on brightness of images
  - Usually image derivatives
  - Popular technique : detector

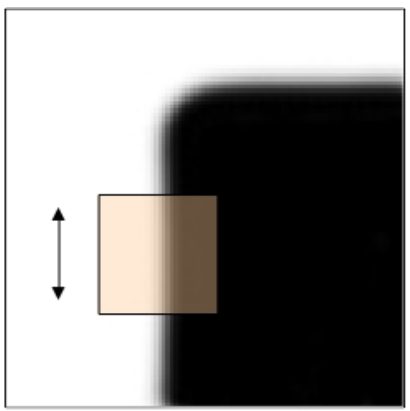
Harris Corner Detector

Basic Idea

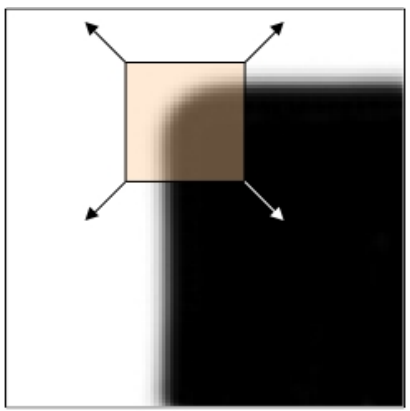
- We should easily recognize the point by looking through a small window
- Shifting a window in any direction should give a large change in intensity



flat region
no change in all directions



edge
no change along the edge direction



corner
significant change in all directions

Harris Corner Detector

Mathematics

- Change in appearance of window W for the shift $[u, v]$:

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

- First-order Taylor approximation

$$I(x + u, y + v) \approx I(x, y) + I_x(x, y)u + I_y(x, y)v$$

- Let's plug this into $E(u, v)$

$$E(u, v) \approx \sum_{(x,y)\in W} [I(x, y) + I_x(x, y)u + I_y(x, y)v - I(x, y)]^2$$

$$\approx \sum_{(x,y)\in W} \left[I_x^2 u^2 + 2I_x I_y uv + I_y^2 v^2\right]$$

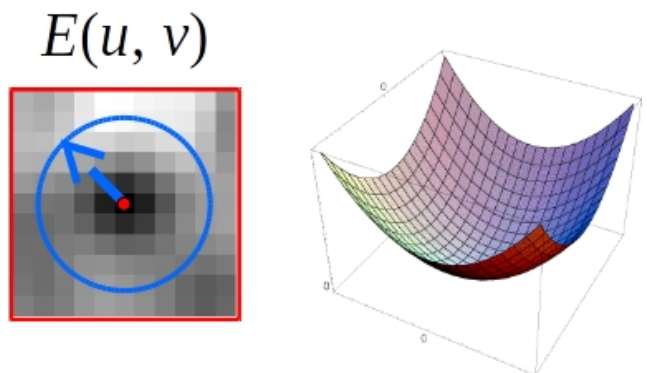
Harris Corner Detector

Mathematics

- Change in appearance of window W for the shift $[u, v]$:

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} \sum_{x,y} I_x^2 & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

- The surface E(u,v) is locally approximated by a quadratic form.



  - In which directions does it have the smallest/greatest change ?

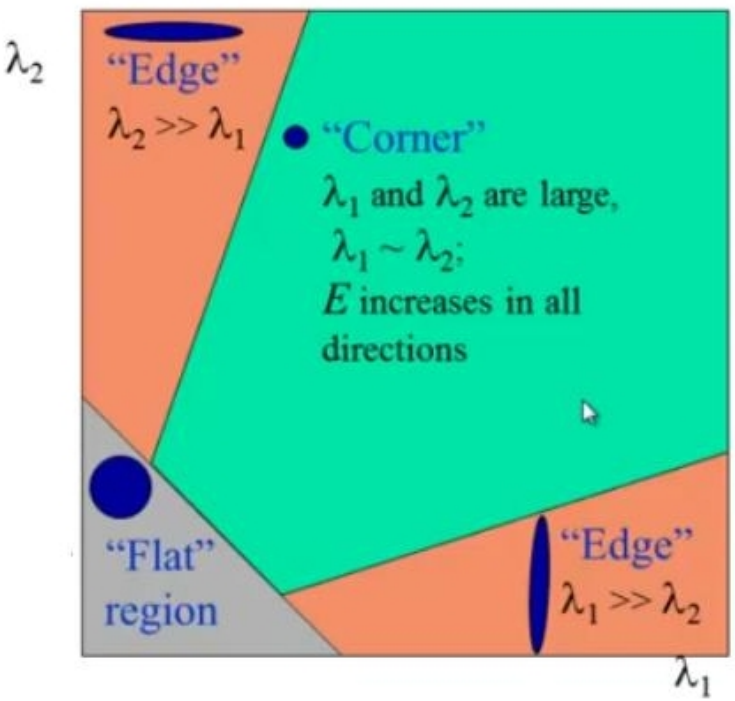
Harris Corner Detector

Harris region

- Matrix and eigenvalues

$$M = \begin{bmatrix} \sum_{x,y} I_x^2 & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y^2 \end{bmatrix}$$

- Analysis of λ_1 and λ_2, the eigenvalues of M



Harris Corner Detector

Criteria

- Harris criterion

$$R = det(M) - k(trace(M))^2$$

  - $|R|$ is small : λ_1 and λ_2 are small $\rightarrow$ the region is flat.
  - $R < 0$: $\lambda_1 >> \lambda_2$ or vice versa $\rightarrow$ the region is edge.
  - R is large : λ_1 and λ_2 are large and $\lambda_1 \sim \lambda_2$, $\rightarrow$ corner.

- Shi-Tomasi criterion : threshold

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

  - Experimentally, this score criteria was much better.

Harris Corner Detector

Algorithm

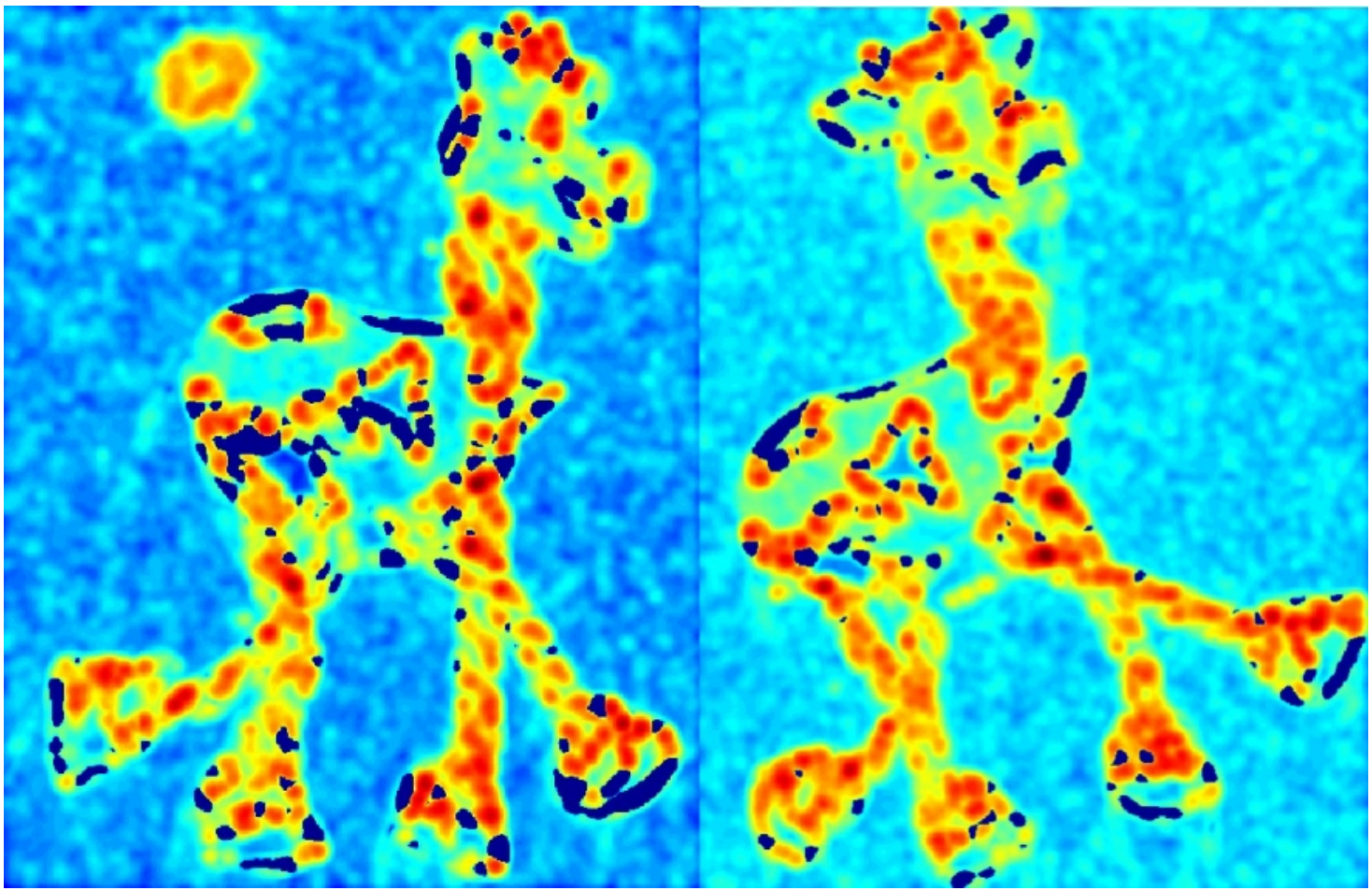
- Compute horizontal and vertical derivatives of image I_x and I_y
- Compute three images corresponding to three terms in matrix M. Convolve these three images with a Gaussian filter.
- Compute R
- Find local maxima by thresholding R and applying non maximum suppression.

Harris Corner Detector

Example

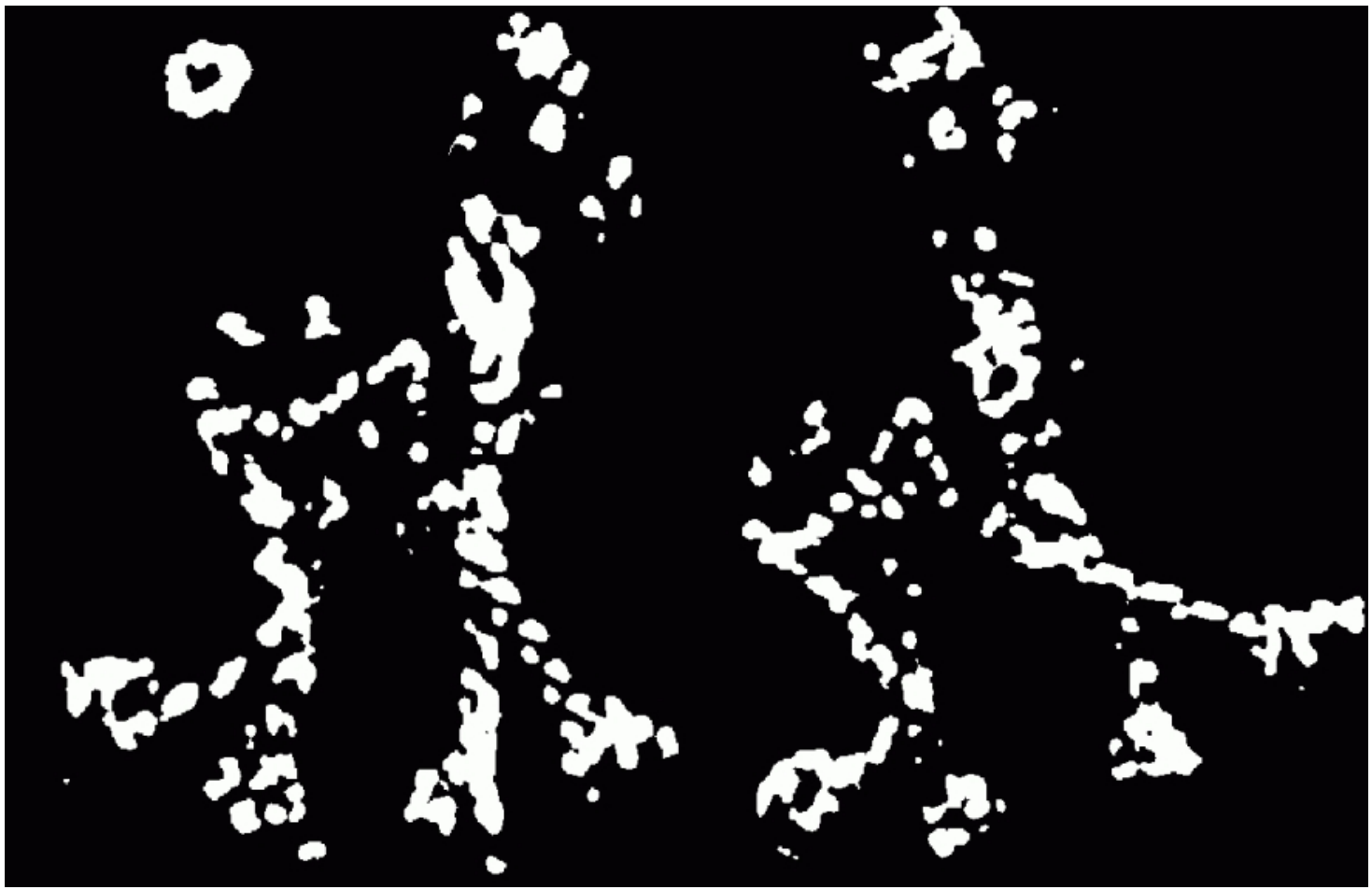
Harris Corner Detector

Example



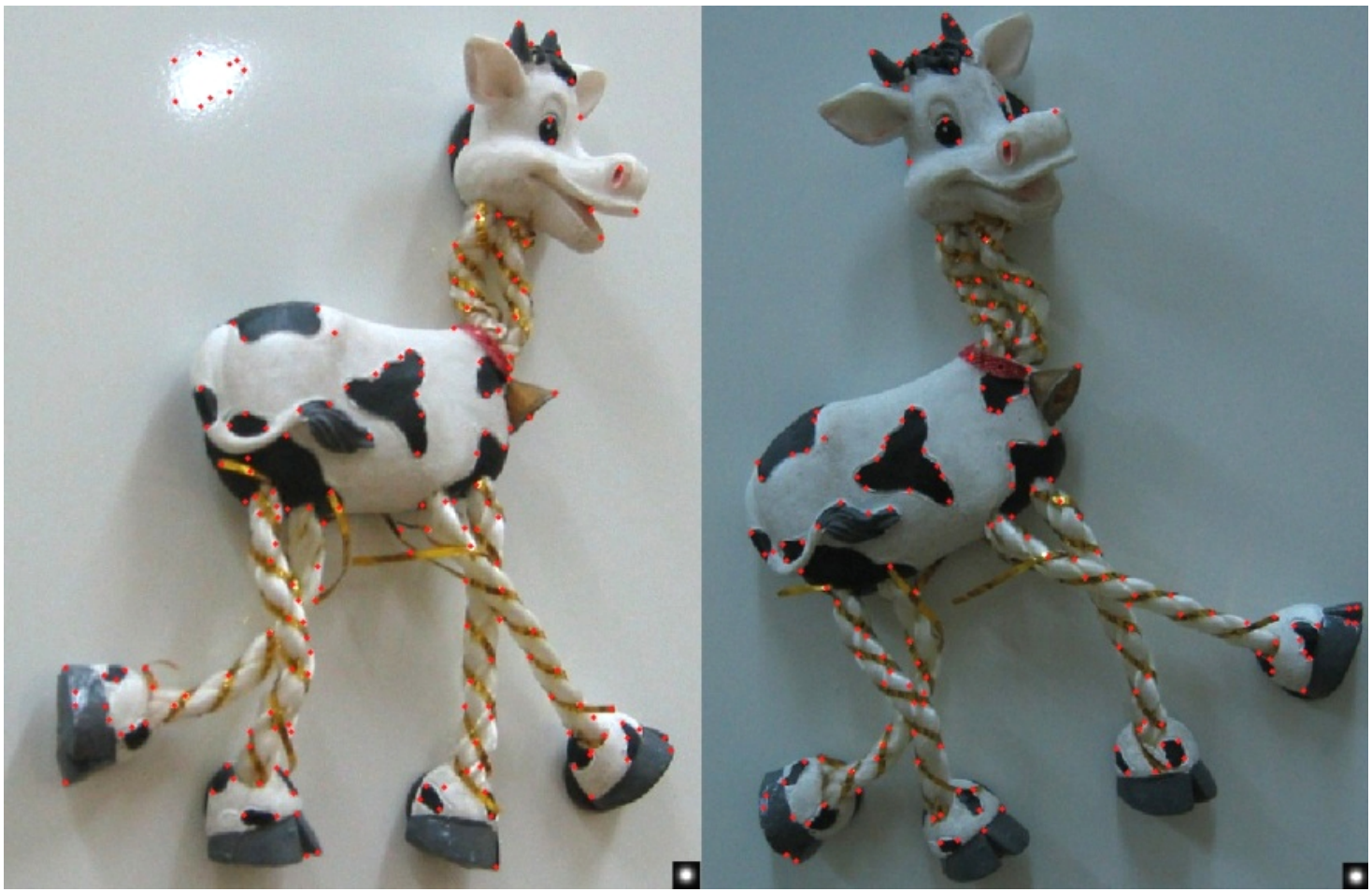
Harris Corner Detector

Example



Harris Corner Detector

Example



Harris Corner Detector

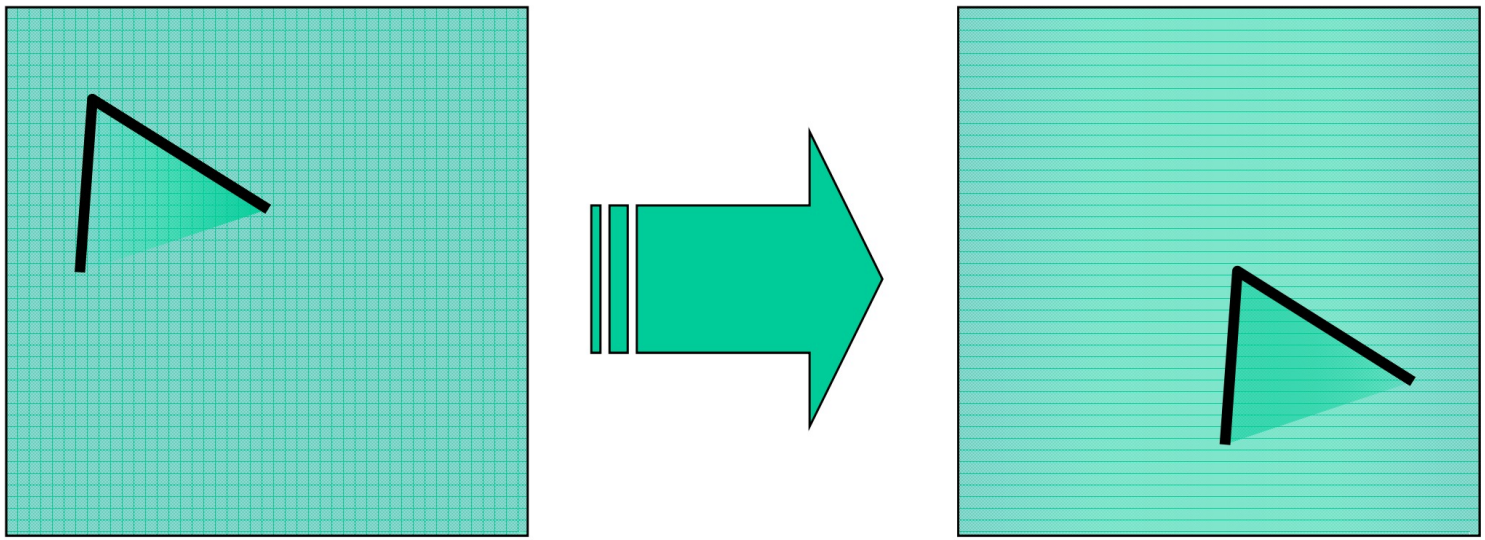
Invariance and covariance

- Invariance : when the image is transformed, corner locations doe not change.
- covariance : when the image is transformed, corner locations get transformed with the same function.

Harris Corner Detector

Invariance and covariance

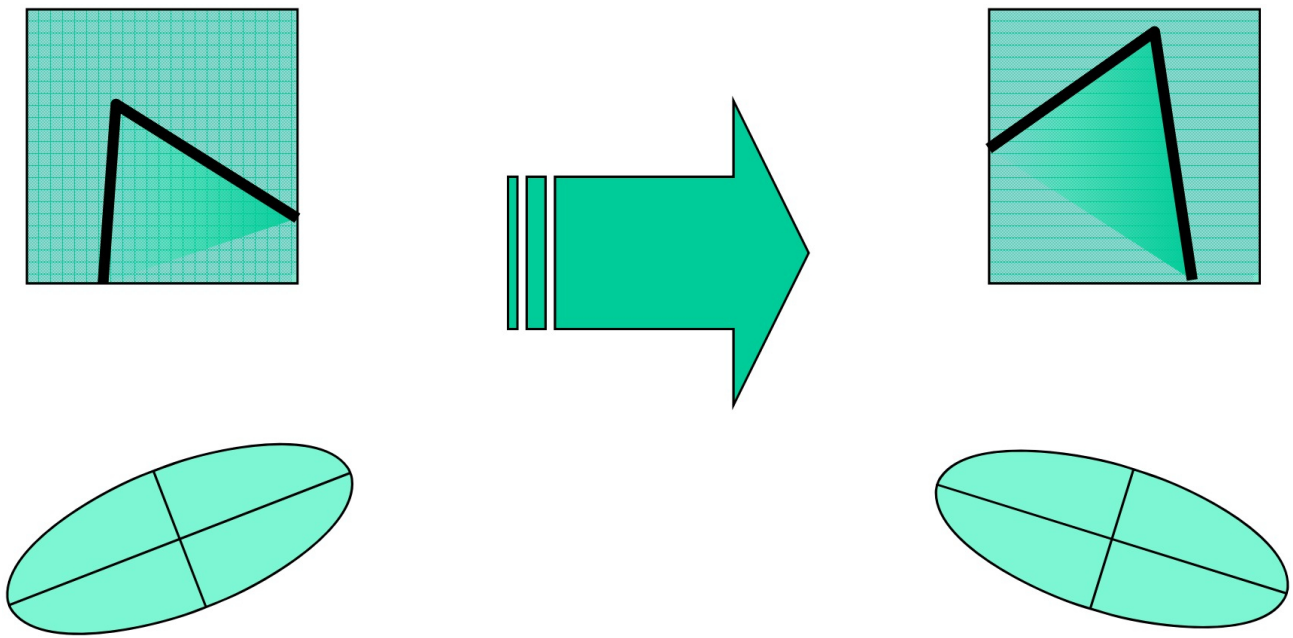
- Corner location is covariant w.r.t. translation



Harris Corner Detector

Invariance and covariance

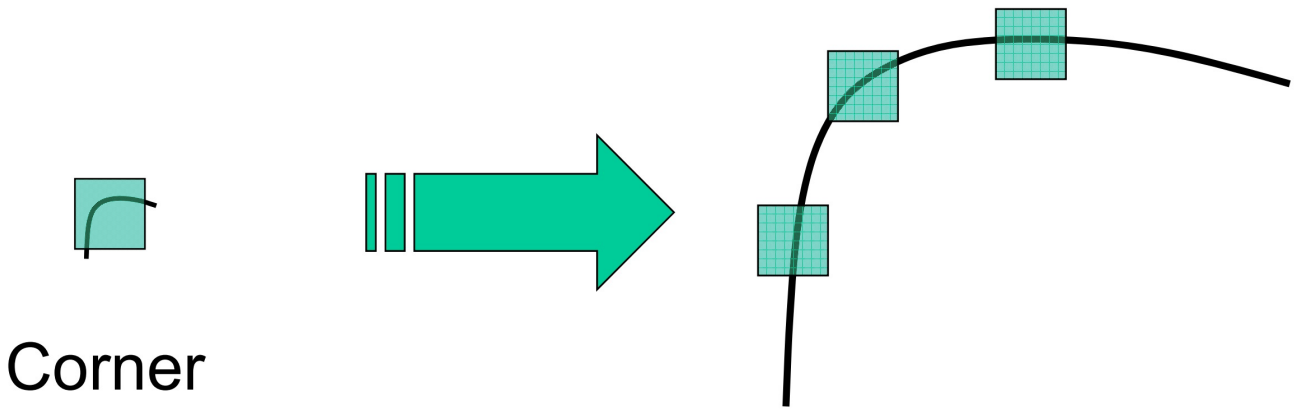
- Corner location is covariant w.r.t. rotation



Harris Corner Detector

Invariance and covariance

- Corner location is not covariant to scaling !



Harris Corner Detector

Invariance and covariance

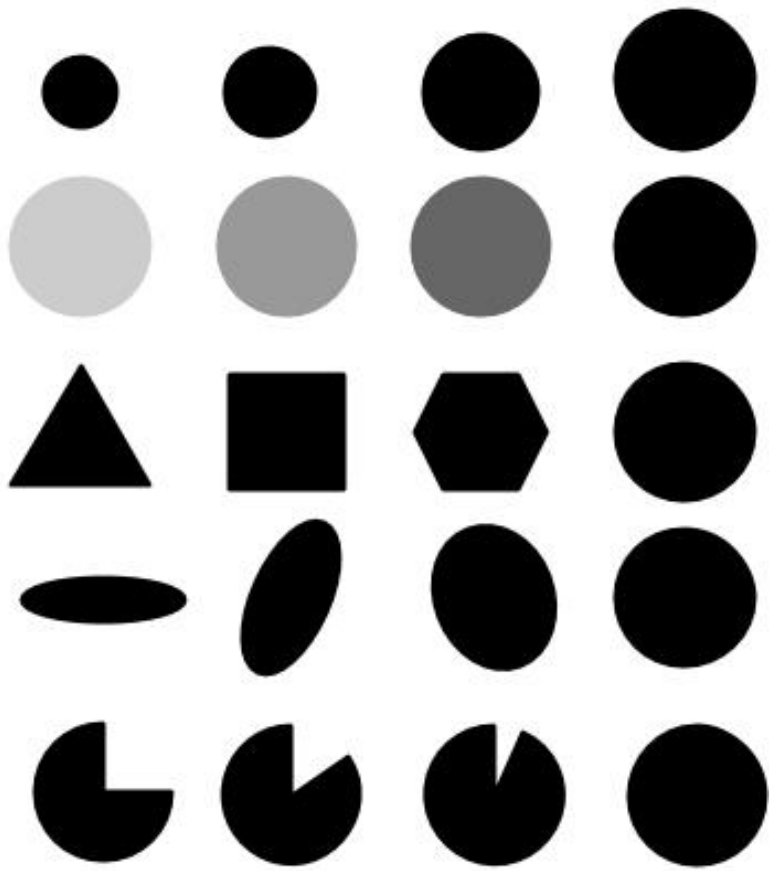
- Affine intensity change : $J = aI + b$
- Only derivatives are used : invariance to intensity shift $J = I + b$
- Intensity scaling : $J = aI$

Corner location is not covariant to scaling !

Blob detection

Simple blob detection

- A Blob is a group of connected pixels in an image that share some common property (E.g grayscale value).

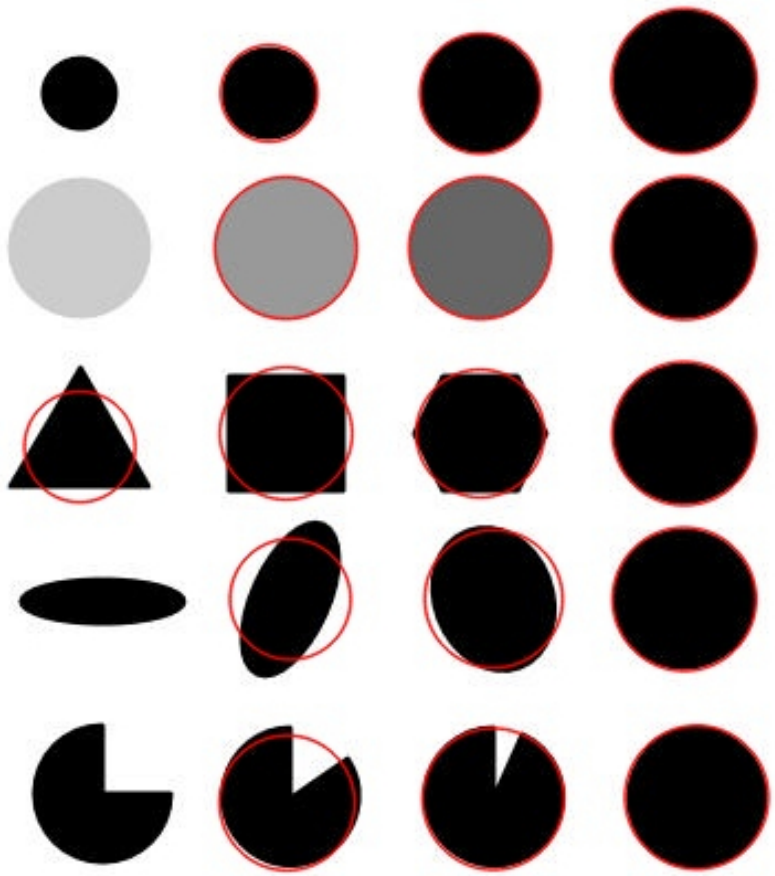


Simple blob detection

- Algorithm
  1. **Thresholding** : Convert the source images to several binary images by thresholding the source image with thresholds : *minThreshold* : *thresholdStep* : *maxThreshold*
  2. **Grouping** : Extract connected components from every binary image and calculate their centers.
  3. **Merging** : Blobs whose centers are located closer than minDistBetweenBlobs are merged.
  4. **Center and Radius Calculation** : The centers and radii of the new merged blobs are computed and returned.

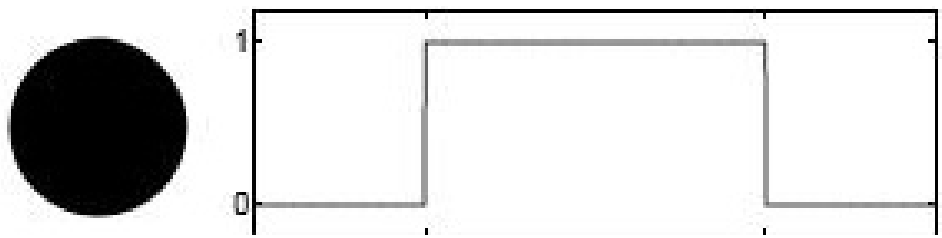
Simple blob detection

- Results



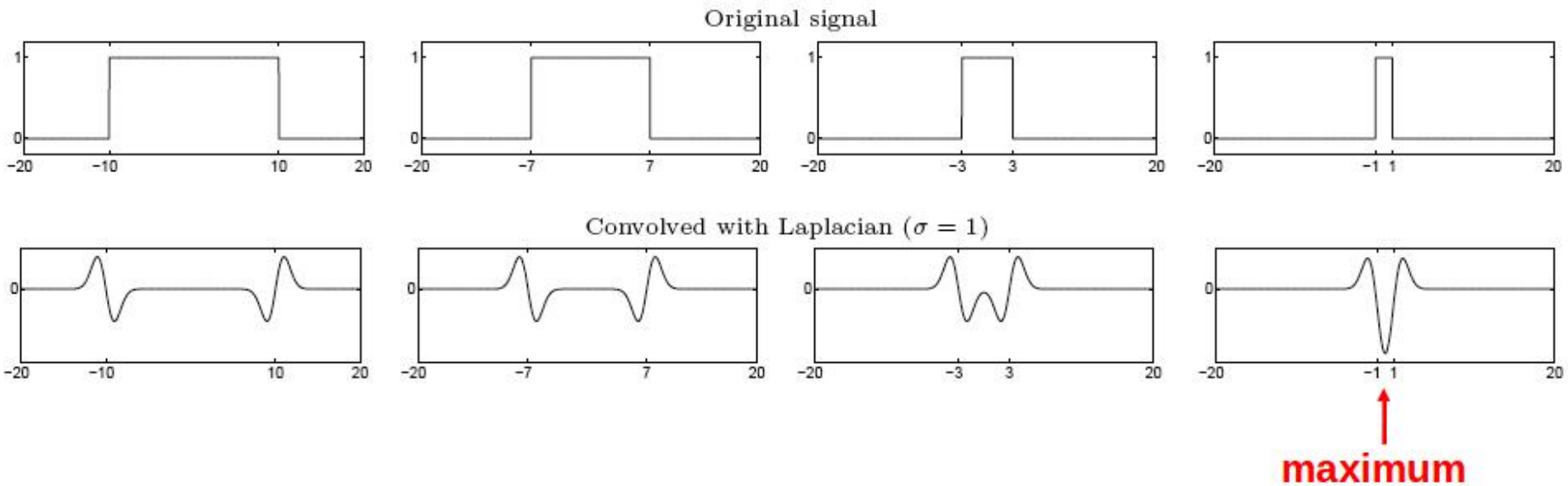
Blob detection

- Blobs are bright on dark or dark on bright regions in an image.
- Blob = superposition of two ripples



Blob detection

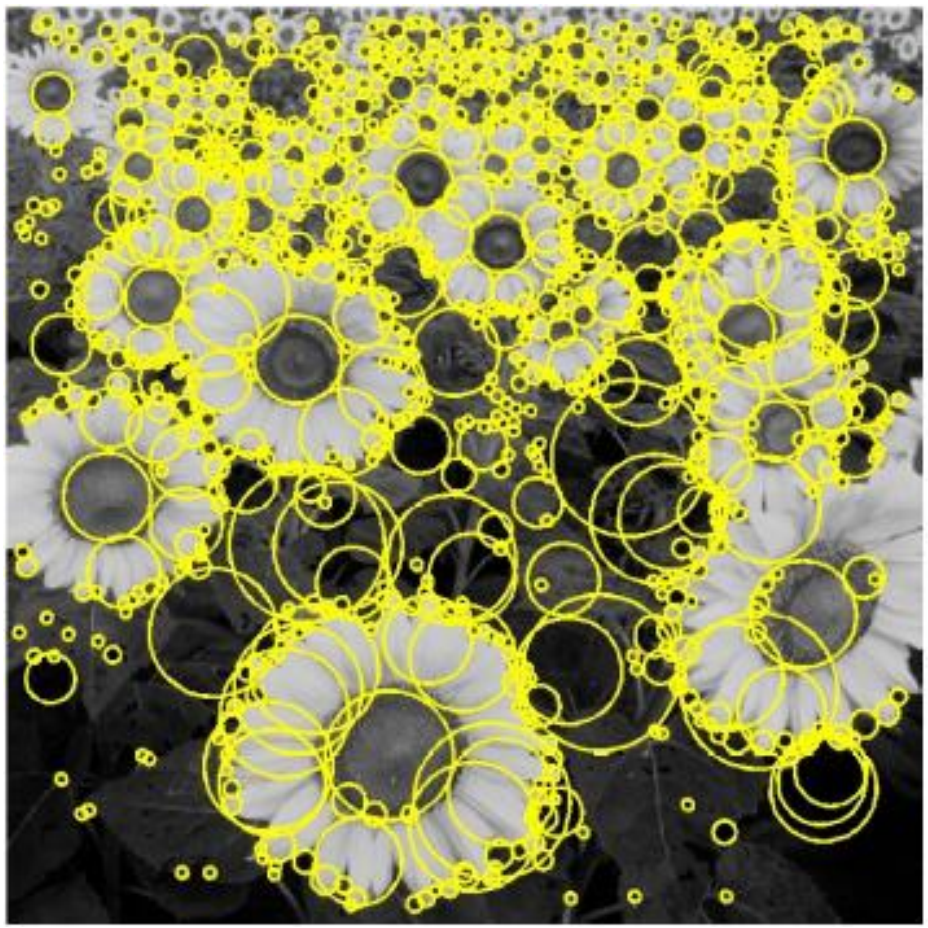
- Convolution of a blob signal with a Laplacian filter $\sigma = 1$

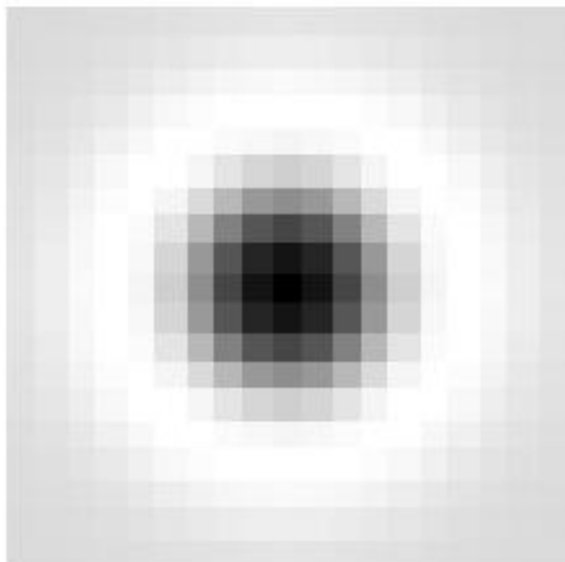


- Matching between the blob size and the Laplacian variance σ^2.

Blob detection

- Blobs with different scales





LoG detector

Expression

- "Scale normalized"-Laplacian of the Gaussian (LoG)

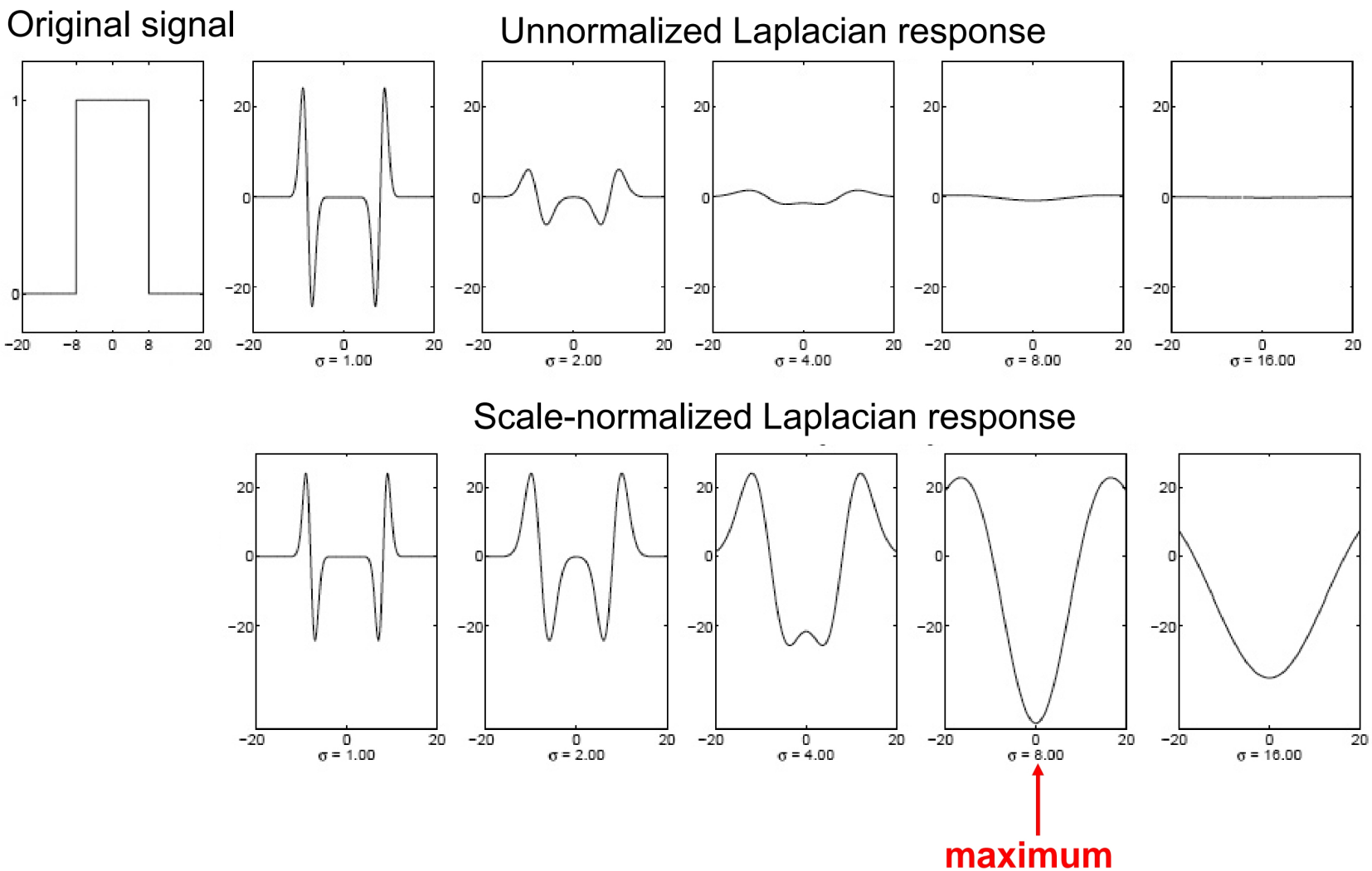
$$L(x, y; \sigma) = \sigma^2 \nabla^2 g(x, y, \sigma) * f(x, y),$$

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

- σ^2 is a scale normalization ; without, $\lim_{\sigma \to +\infty} L(x, y; \sigma) = 0$
- Strong positive responses for dark blobs (maximum response for $\sigma = r/\sqrt{2}$).
- Strong negative responses for bright blobs of similar size.
- For a given blob, choose the right scale for the right blob size.

LoG detector

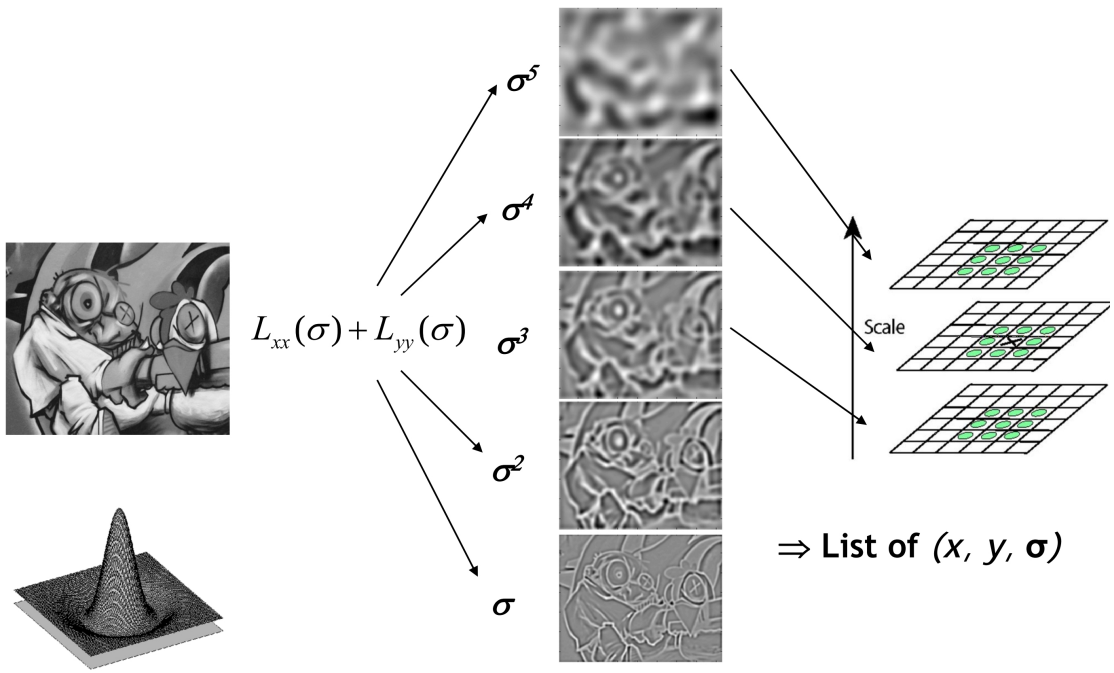
Effect of Scale normalization



LoG detector

Scale selection [1]

- Convolve the input with scale-normalized LoG at several scales.

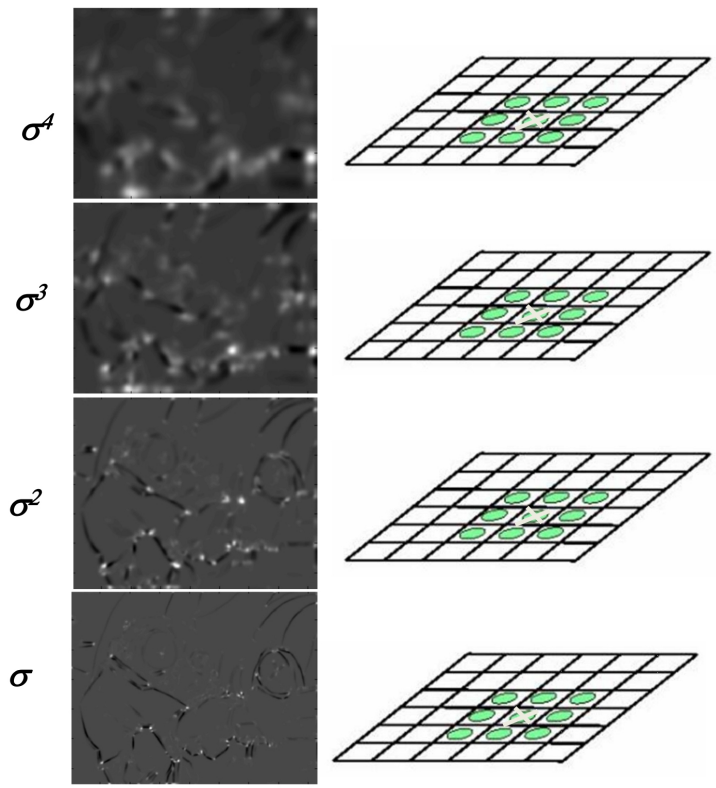


- Find local maxima of squared Laplacian response in scale-space
  - Select (x, y, σ) that maximizes $NormalizedLoG(x, y, \sigma)$.

1. T. Lindeberg. Feature detection with automatic scale selection. IJCV, 1998.

Harris-Laplace Detector [2]

- Multiscale Harris corner detection : $\sigma_n = s^n \sigma_0$.

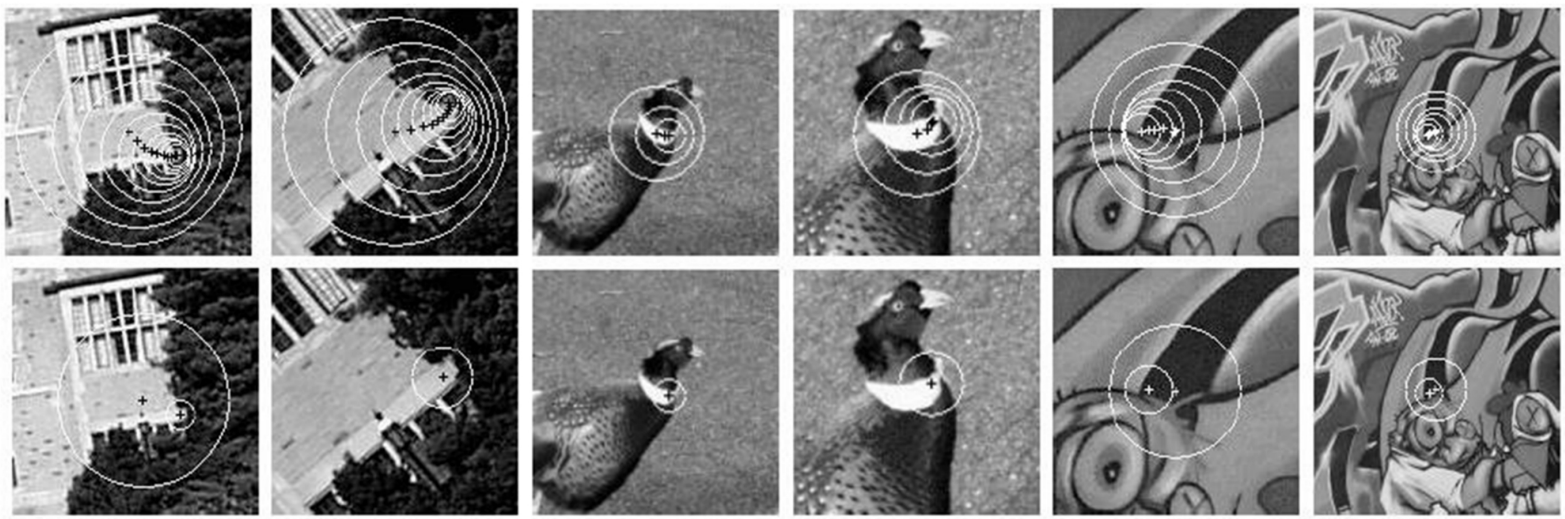


2. K.Mikolajczyk, C.Shmitd "Indexing Based on Scale invariant Interest Points", ICCV 2001.

Harris-Laplace Detector [2]

- Scale selection based on Laplacian

Harris points



Harris-Laplace points

2. K.Mikolajczyk, C.Shmitd "Indexing Based on Scale invariant Interest Points", ICCV 2001.

SIFT

Basic idea

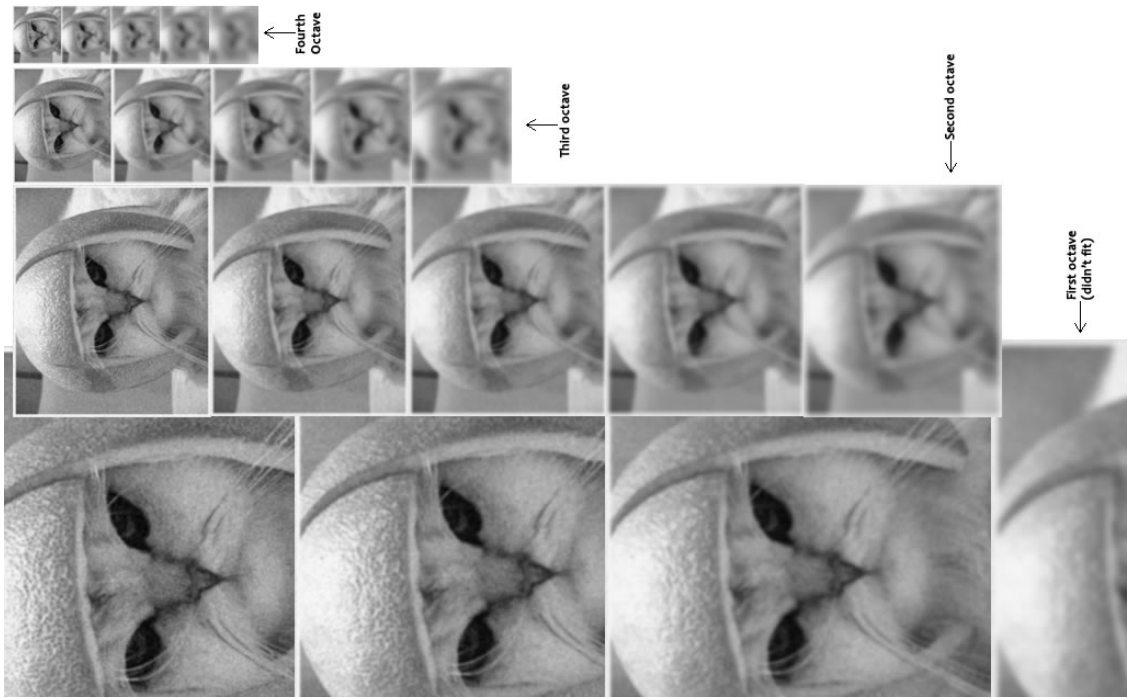
- In the Scale-space, select (x, y, σ) that maximizes $DoG(x, y, \sigma)$ (approx. of approximation of LoG) in all three dimensions.
- Use of an efficient implementation.
- Keypoints description : Definition of a feature vector for each keypoint.

Algorithm

1. The scale space.
2. LoG approximations : DoG.
3. Finding keypoints.
4. Getting rid of low contrast keypoints.
5. Keypoint orientations.
6. Generating a feature.

Algorithm

- **1. Scale space** : repeating generating blurred half-sized images.



- Images of the same octave have same size.
- Each octave has 5 images obtained by increasing the blur.

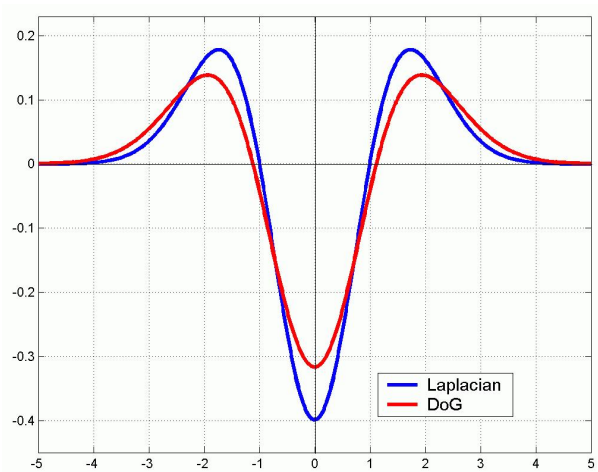
Algorithm

- **1. Scale space** : repeating generating blurred half-sized images.
- Image size reduction
  - Reason 1 : when σ increases, the kernel size needed for precise calculations increases, and so do the calculation time. We reduce the image size instead of increasing σ.
  - Reason 2 : After the convolution with the Gauss kernel, the higher frequencies in the image spectrum are almost erased - there is no gain in keeping a higher resolution

Algorithm

- **2. LoG approximations** of the Laplacian with a difference of Gaussians

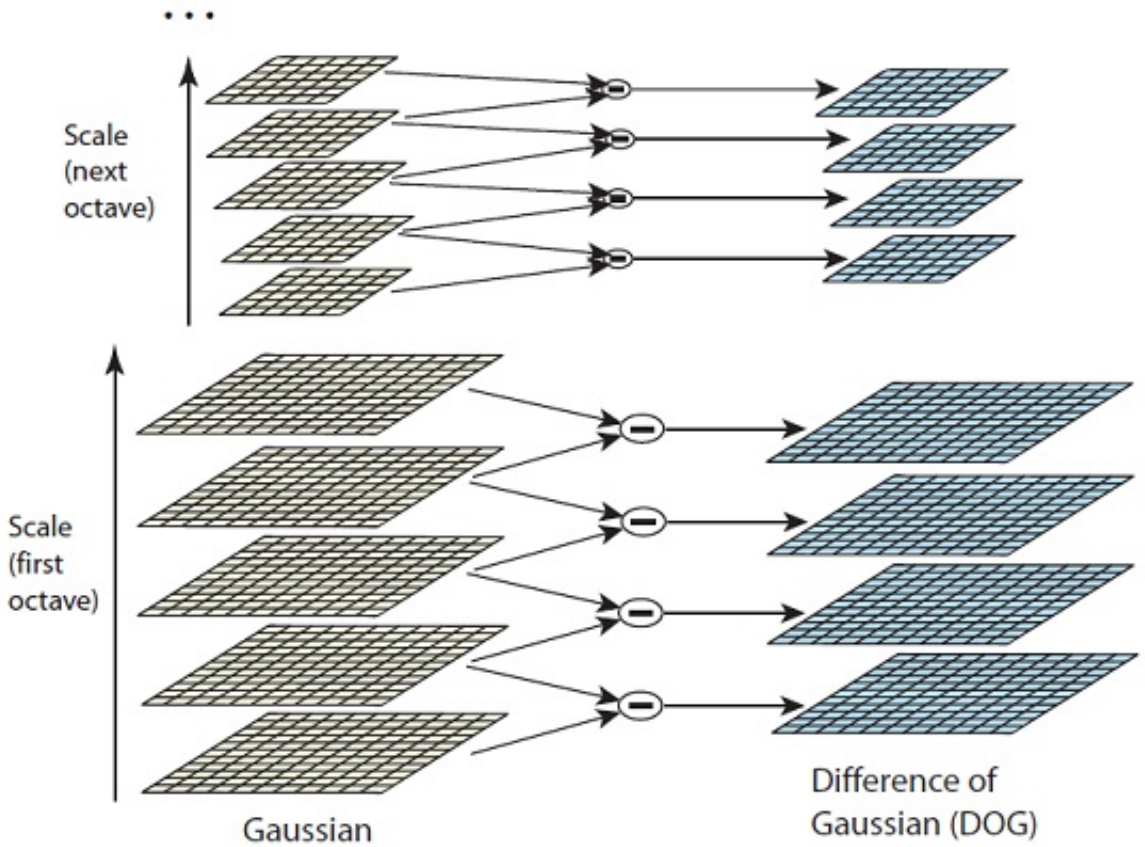
$$DoG = G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2 \Delta G(x, y)$$



- Typical values : $\sigma = 1.6$, $k = \sqrt{2}$.
- *DoG* already incorporate the σ^2 normalization.

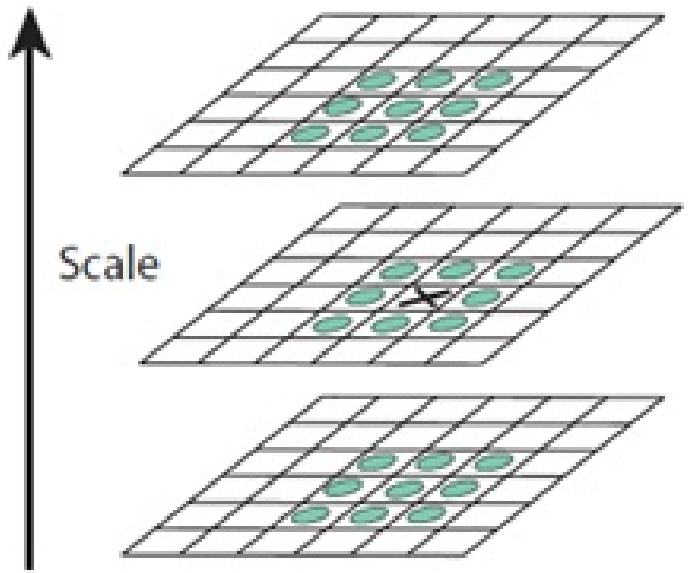
Algorithm

- **2. LoG approximations** of the Laplacian with a difference of Gaussians



Algorithm

- **3. Finding keypoints** : For each octave, iterate over the pixels. X is marked as a "key point" if its DoG is the greatest or least of all 26 neighbors.

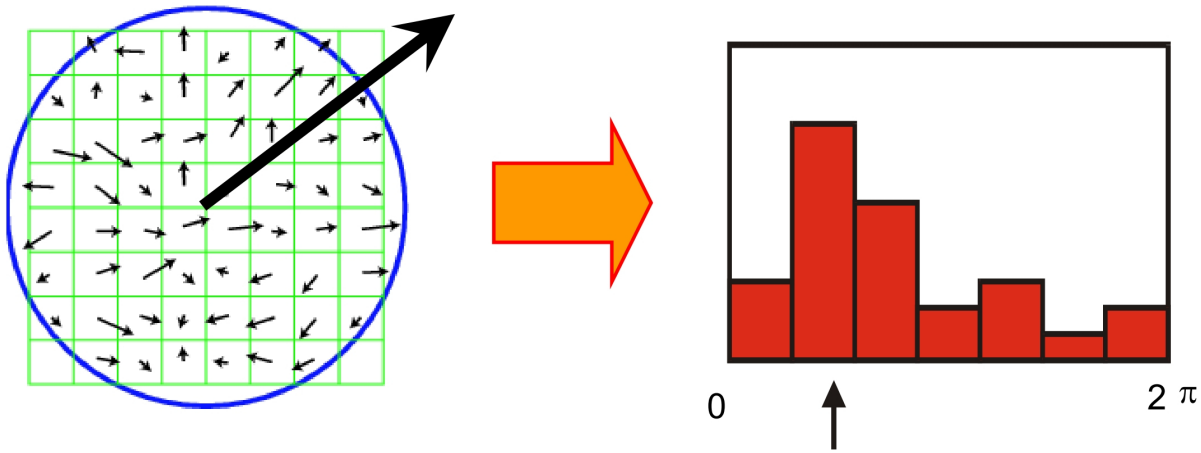


Algorithm

- **4. Getting rid of low contrast keypoints**
- Removing low contrast features in the DoG image.
- Removing edges : use the Harris corner detector.

Algorithm

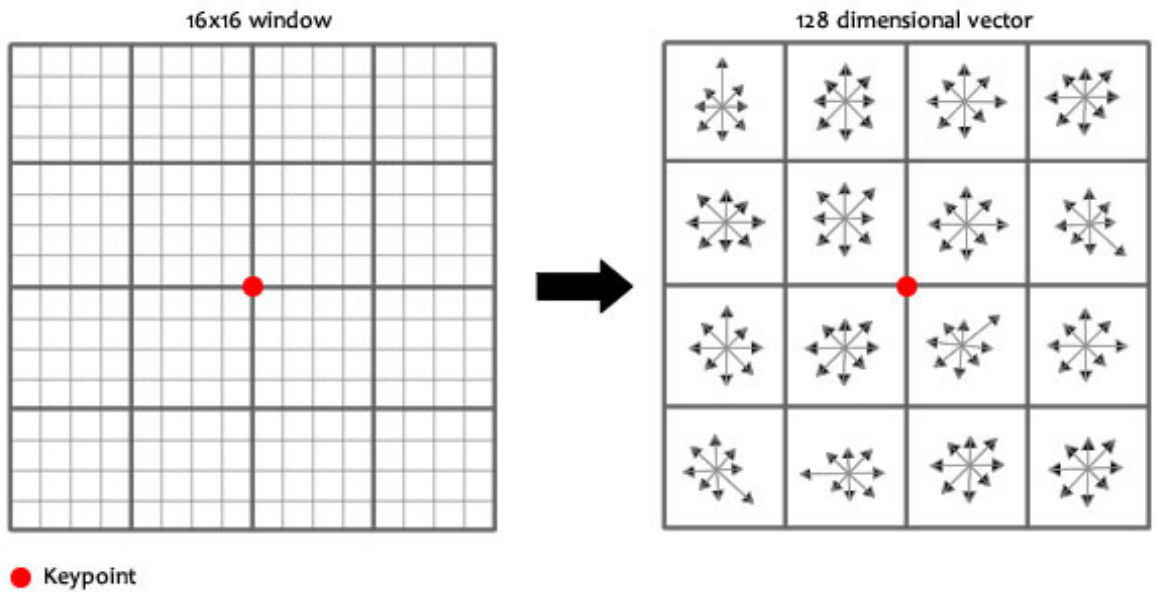
- **5. Keypoint orientations**
- To assign a unique orientation to circular image windows :
  - Create histogram of local gradient directions in the patch
  - Assign canonical orientation at peak of smoothed histogram



- Any later calculations are done relative to this orientation. This ensures rotation invariance.

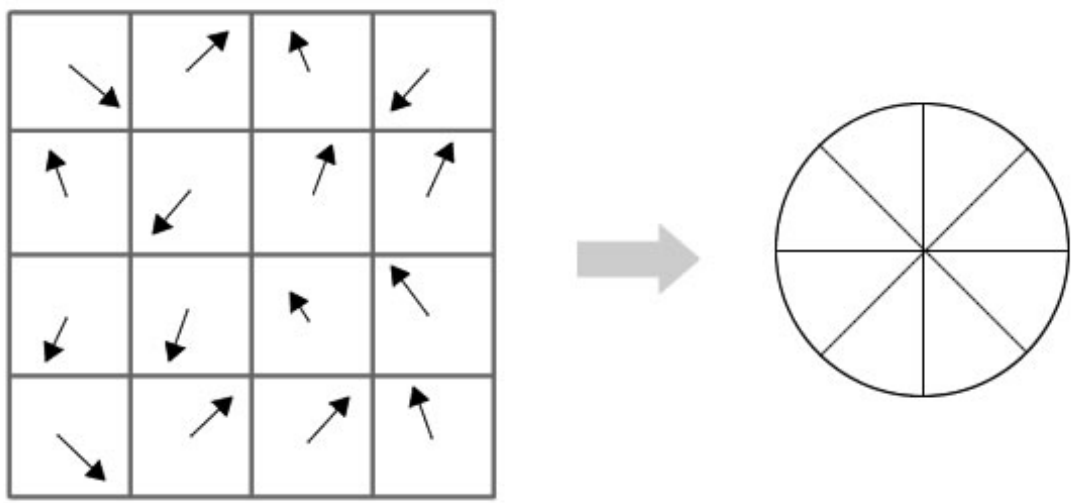
Algorithm

- **6. Generating a feature vector** for the keypoint.
- A 16x16 window around the keypoint. This 16x16 window is broken into sixteen 4x4 windows.



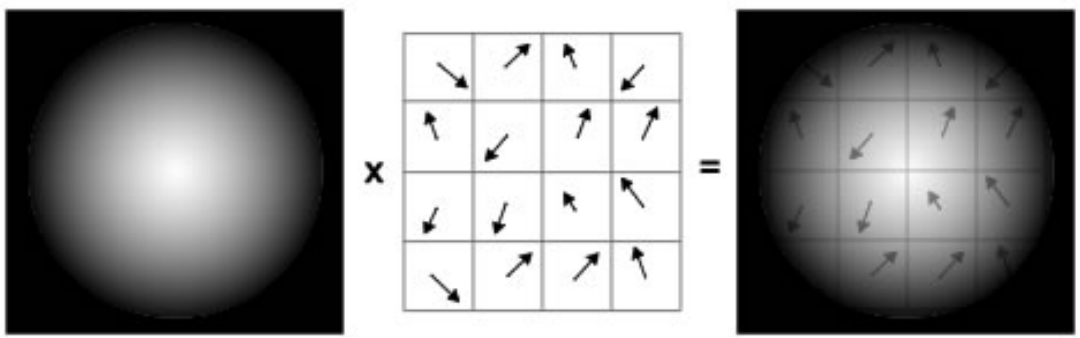
Algorithm

- **6. Generating a feature vector** for the keypoint.
- Within each 4x4 window, gradient magnitudes and orientations are calculated. These orientations are put into an 8 bin histogram.



Algorithm

- **6. Generating a feature vector** for the keypoint.
- Form weighted histogram (8 bin) for 4x4 regions
  - Weight by magnitude and spatial Gaussian



- Concatenate 16 histograms in one long vector of 128 dimensions.