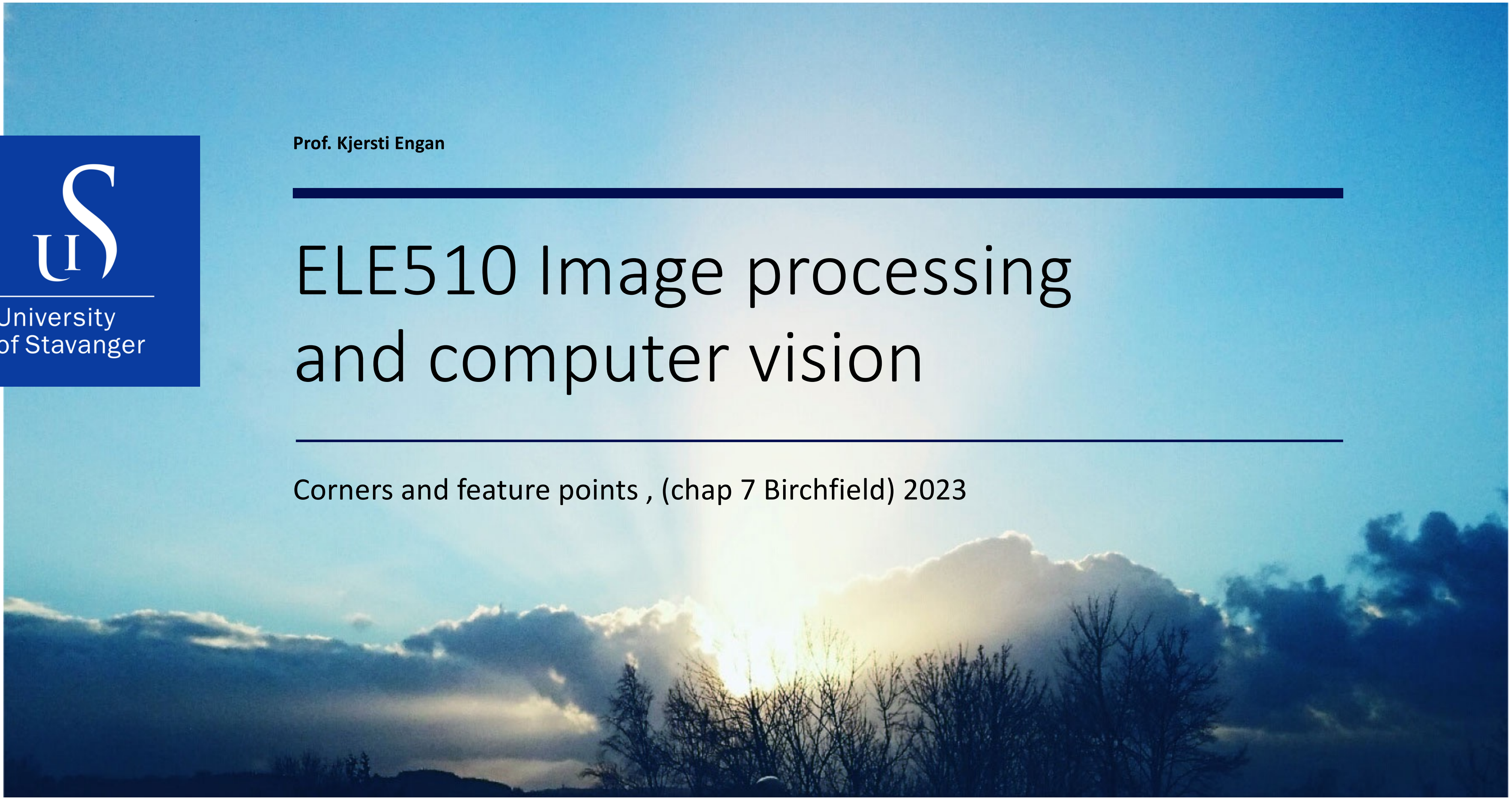


University of Stavanger

Prof. Kjersti Engan

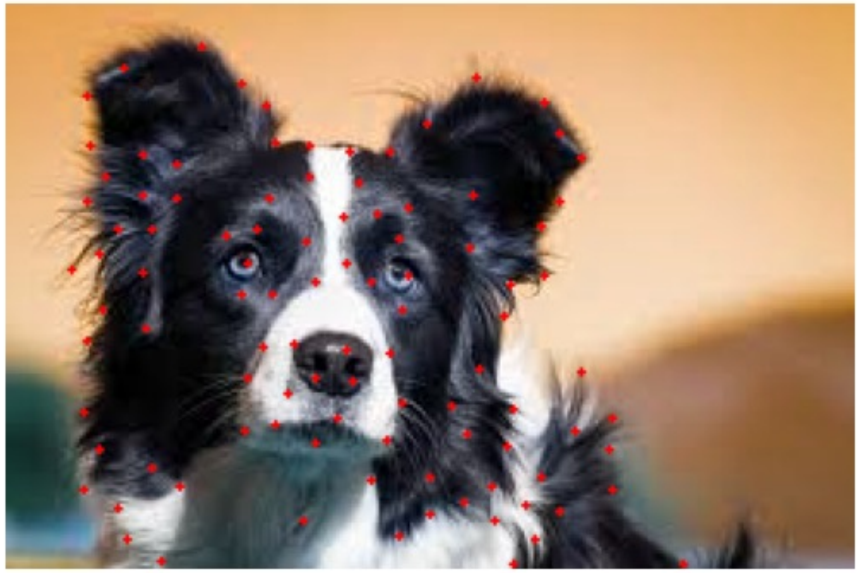
# ELE510 Image processing and computer vision

Corners and feature points , (chap 7 Birchfield) 2023

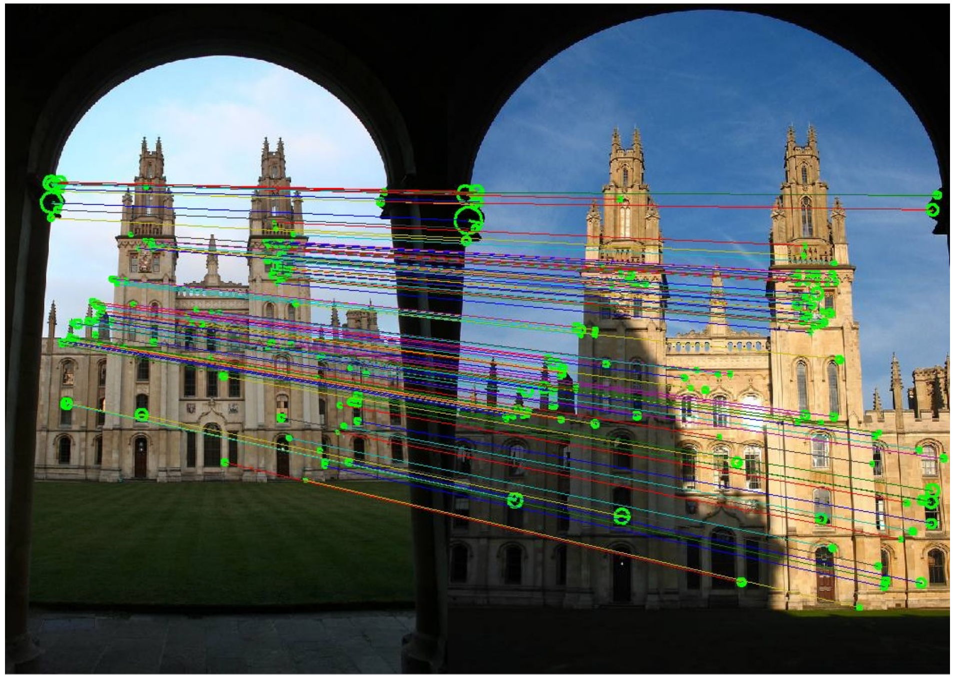
## Lines, corners and feature points in images

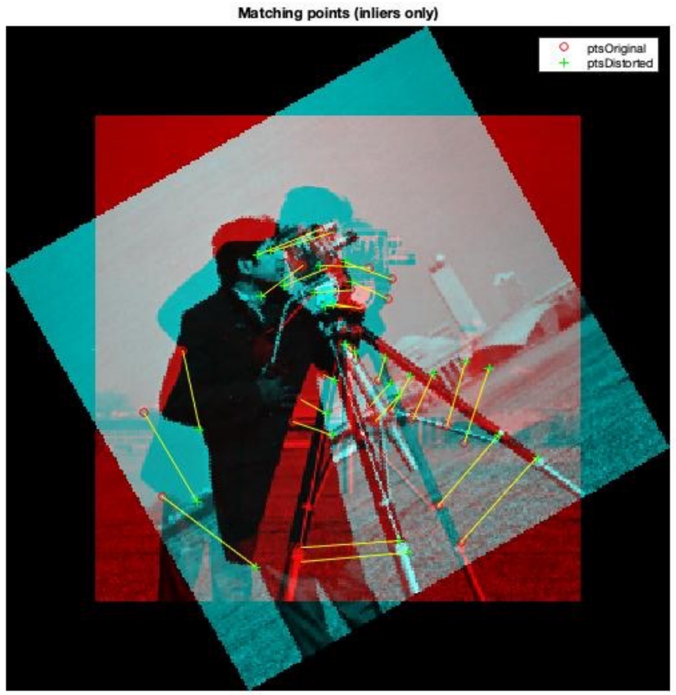
- Three  points from the topic:

1. What are important (corner) points in an image?
2. How can we make features from corner-points?
3. Some applications that use corner-points and features

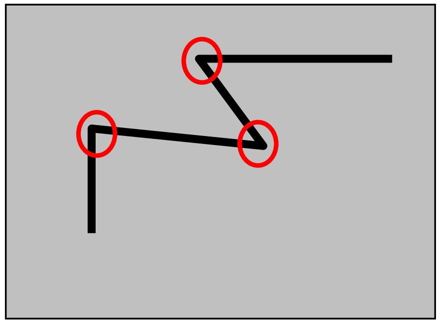


## Lines and feature points – describe your image





## (7.4) Feature detectors

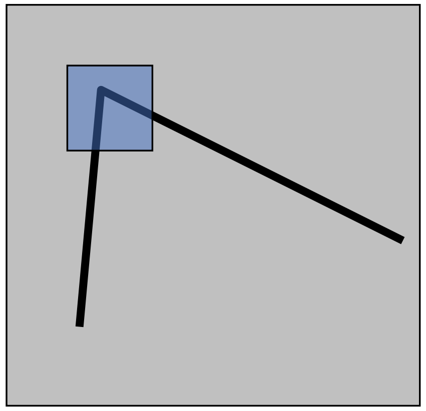


- An *edge detector* finds pixels where the magnitude of the gradient is large.
- A *feature point detector* ( or interest operator) seeks pixels where the graylevel values vary locally in more than one direction.
- **Feature points, interest points or corner points** are unique and distingusihable from other pixels using information in the local neighborhood.
- Can be used to
  - find/classify objects,
  - align multiple images,
  - image registration
  - camera calibration,
  - stereo imaging,
  - ...

## Harris Corner Detector: Main idea

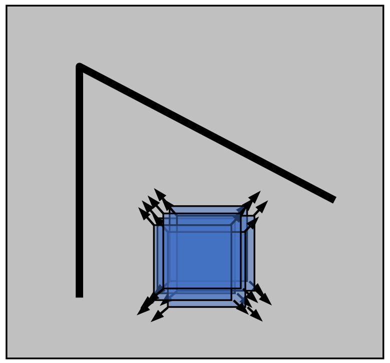
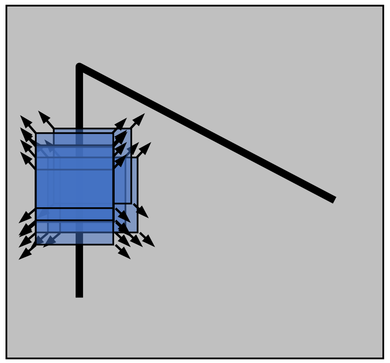
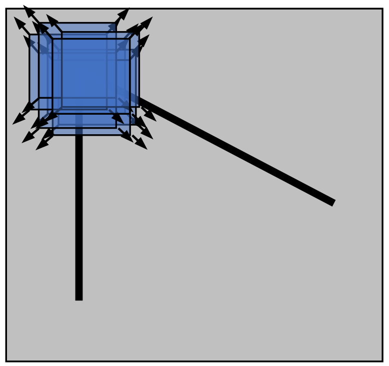
- The most famous ***corner detector*** is the *Harris corner detector* after a work of Chris Harris and Mike Stephens from 1988.

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



Some of the illustrations used on Harris corner detector is taken from a presentation based on slides from Rick Szeliski's lecture notes, CSE576, Spring 2005.

# The Harris corner detector

"flat" region:
no change in
all directions

"edge":
no change along
the edge direction

"corner":
significant change
in all directions

# Harris Detector: Mathematics

I(x,y) is image, w(x,y) is local window.
Change of intensity for the shift [*u,v*], E(u,v):

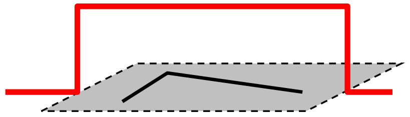
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

Window function

Shifted intensity

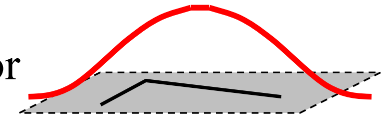
Intensity

Window function $w(x,y) =$



1 in window, 0 outside

or



Gaussian

I(x+u,y+v) can be approximated using a Taylor series.

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

Define image gradients :
These can be found by ex. Sobel

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

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

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

1. Order Taylor approximation:

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

Remember, I(x,y) is image, w(x,y) is local window. Change of intensity for the shift [*u,v*]

$$\boxed{E(u,v) = \sum_x \sum_y w(x,y)[I(x+u,y+v) - I(x,y)]^2}$$

Now, for small shifts (u,v) we use this bilinear approximation (1. order Taylor, last slide):

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

$$E(u,v) \approx \sum_x \sum_y w(x,y)[I(x,y) + I_x(x,y)u + I_y(x,y)v - I(x,y)]^2$$

$$\boxed{E(u,v) \approx \sum_x \sum_y w(x,y)[I_x^2(x,y)u^2 + 2I_x(x,y)I_y(x,y)uv + I_y^2(x,y)v^2]}$$

Express in matrix form?

## Harris Detector: Mathematics

For small shifts $[u,v]$ we have a bilinear approximation:

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

where $M$ is a 2×2 matrix computed from image derivatives:

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

Measure for average change in image intensity for small displacements ( other formulation):

$$
\begin{aligned}
I(\mathbf{u}+\mathbf{d}) &\approx I(\mathbf{u}) + \mathbf{d}^T \nabla I \quad \text{for small displacements } \mathbf{d} \\
E(\mathbf{d}) &= \sum_{\mathbf{u}\in \mathcal{W}} w(\mathbf{u}) \left[ I(\mathbf{u}+\mathbf{d}) - I(\mathbf{u}) \right]^2 \approx \sum_{\mathbf{u}\in \mathcal{W}} w(\mathbf{u}) \left[ \mathbf{d}^T \nabla I \right]^2 \\
&\approx \sum_{\mathbf{u}\in \mathcal{W}} w(\mathbf{u}) \mathbf{d}^T \nabla I \nabla I^T \mathbf{d} = \mathbf{d}^T \left[ \sum_{\mathbf{u}\in \mathcal{W}} w(\mathbf{u}) \nabla I \nabla I^T \right] \mathbf{d} \\
&\approx \mathbf{d}^T \mathcal{M} \mathbf{d} \quad \text{where} \quad \mathcal{M} = \sum_{\mathbf{u}\in \mathcal{W}} w(\mathbf{u}) \nabla I \nabla I^T .
\end{aligned}
$$

Harris and Stephens used the properties of the matrix M in the detection of corner points. This matrix M is determined solely by first partial derivatives of the image in a spatial local window, and M is called the **gradient covariance matrix.**

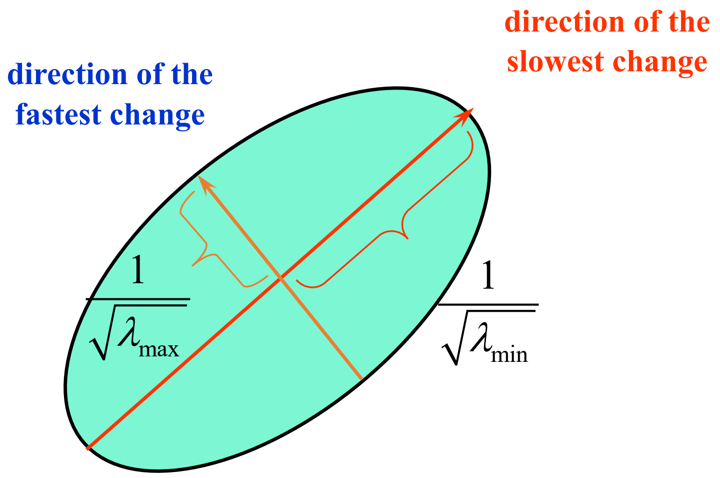
## Properties of the gradient covariance matrix, M

$$\mathcal{M} = \sum_{\mathbf{u}\in\mathcal{W}} w(\mathbf{u}) \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix} \quad \text{where} \quad I_x = \frac{\partial I}{\partial x}, \quad I_y = \frac{\partial I}{\partial y}.$$

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

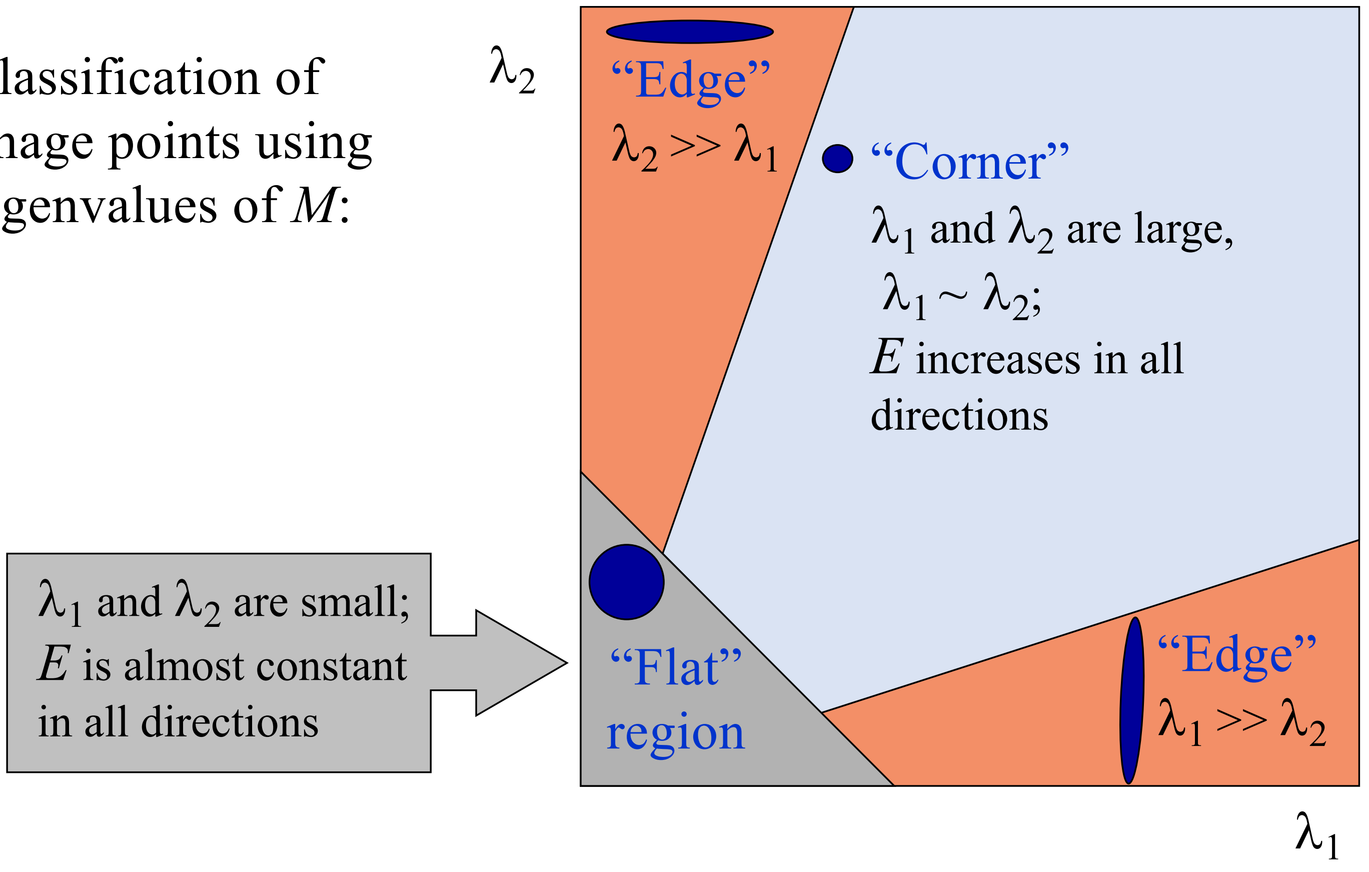
The properties of this matrix can be described by its eigenvalues : $\lambda_1 , \lambda_2$
These can be found by SVD decomposition

- The first eigenvector (the one whose corresponding eigenvalue has the largest absolute value) is the direction of greatest curvature.
- The second eigenvector (the one whose corresponding eigenvalue has the smallest absolute value) is the direction of least curvature.
- The corresponding eigenvalues are the respective amounts of these curvatures.


direction of the fastest change
direction of the slowest change
$\frac{1}{\sqrt{\lambda_{\max}}}$
$\frac{1}{\sqrt{\lambda_{\min}}}$

# Harris Detector: Mathematics

Classification of image points using eigenvalues of $M$:



$\lambda_2$
"Edge"
$\lambda_2 >> \lambda_1$
"Corner"
$\lambda_1$ and $\lambda_2$ are large,
$\lambda_1 \sim \lambda_2$;
$E$ increases in all directions
$\lambda_1$ and $\lambda_2$ are small;
$E$ is almost constant in all directions
"Flat" region
"Edge"
$\lambda_1 >> \lambda_2$
$\lambda_1$

## Harris Detector: Mathematics

Can we avoid doing a SVD decomposition to find eigenvalues explicitly?

Harris and Stephens used the following measure of corner response:

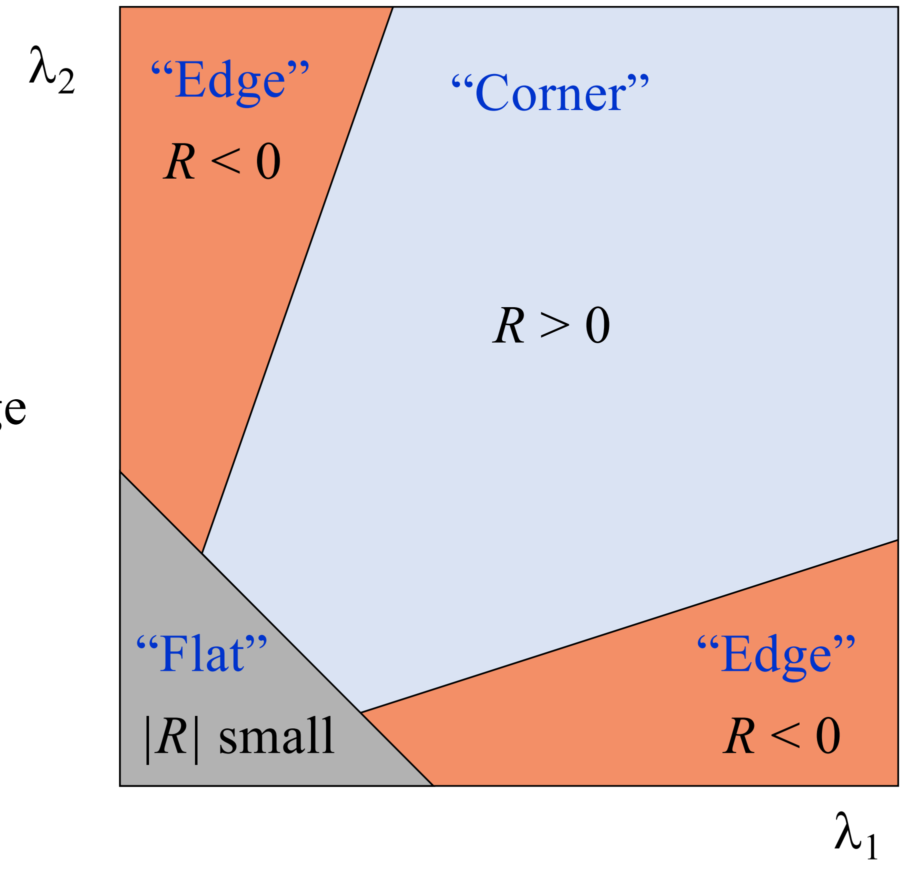
$$R = \det M - k\left(\operatorname{trace} M\right)^2$$

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

*k* is an empirical constant for the “corner point” given by the matrix M and its eigenvalues. The value of *k* is balancing between edge-like and corner-like structures. Typical values for corners are in the range (0.04 – 0.06).

# Harris Detector: Mathematics

- *R* depends only on eigenvalues of M
- *R* is large for a corner
- *R* is negative with large magnitude for an edge
- |*R*| is small for a flat region



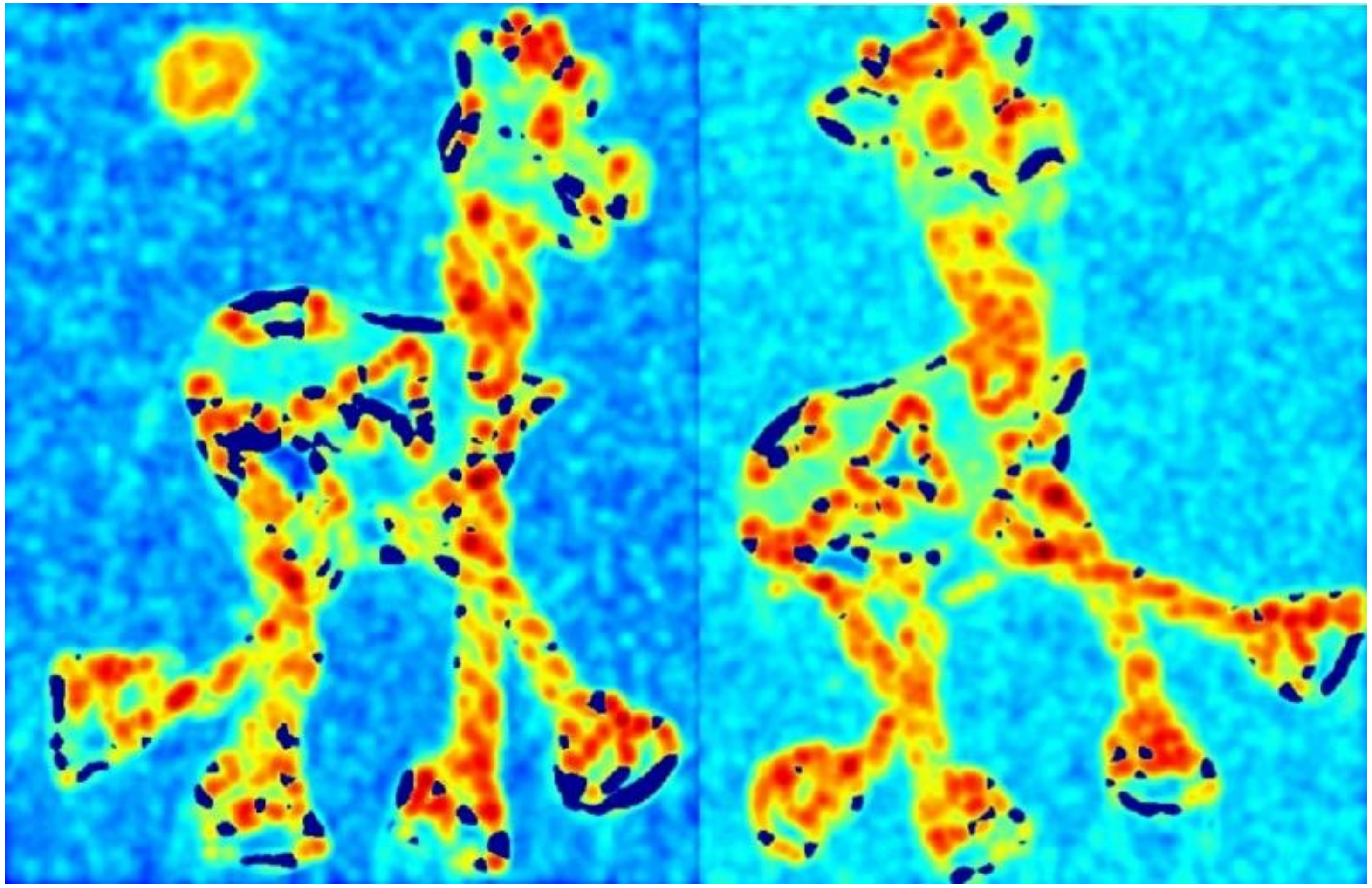
## Harris detector

- The Algorithm:
  - Compute *R*(x,y), corner respons, for all pixel positions.
  - Find points with large corner response function *R* (*R* > threshold)
  - Take the points of local maxima of *R (and R > threshold)*

## Harris Detector: Workflow

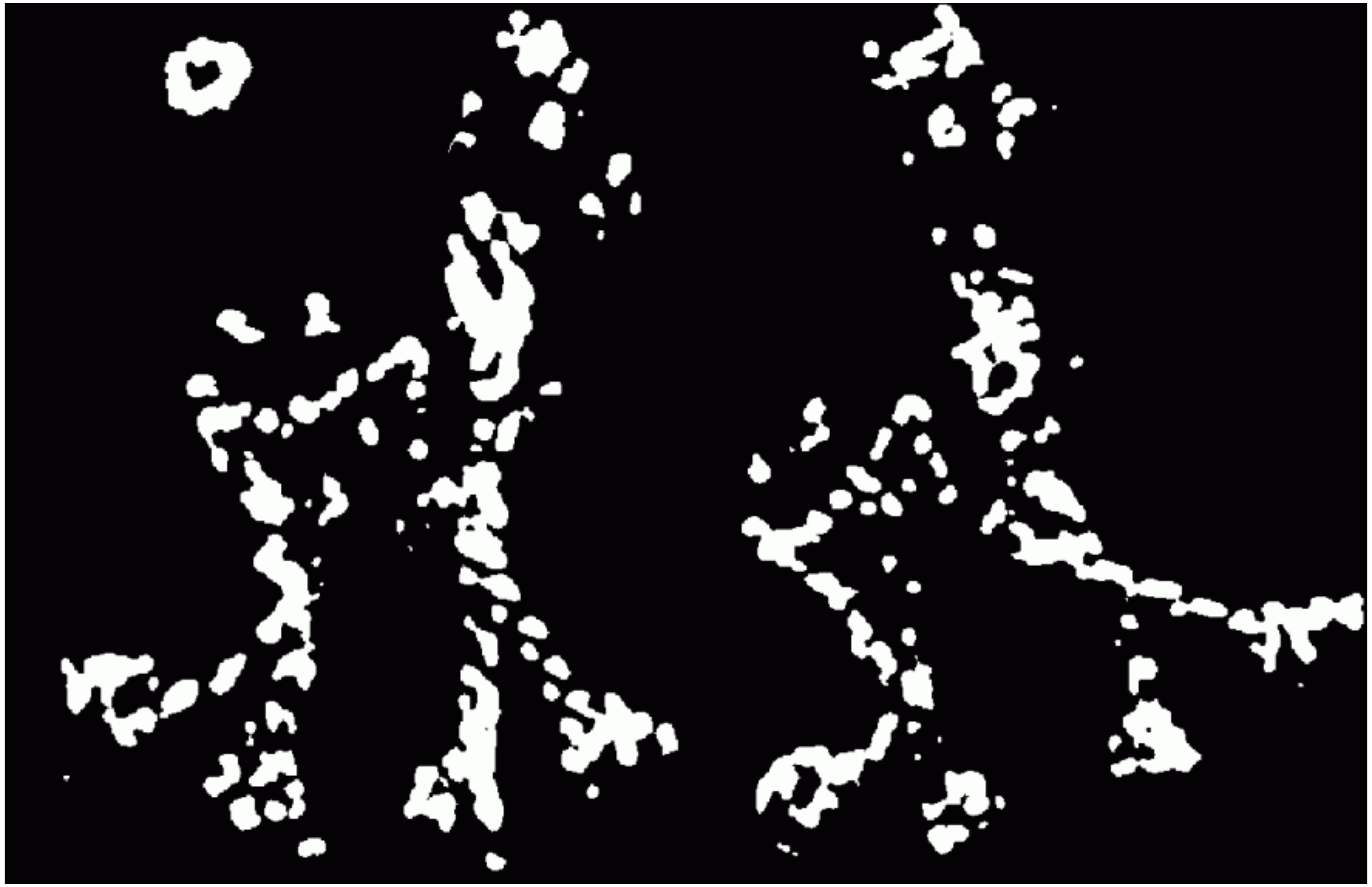
## Harris Detector: Workflow

Compute corner response $R$



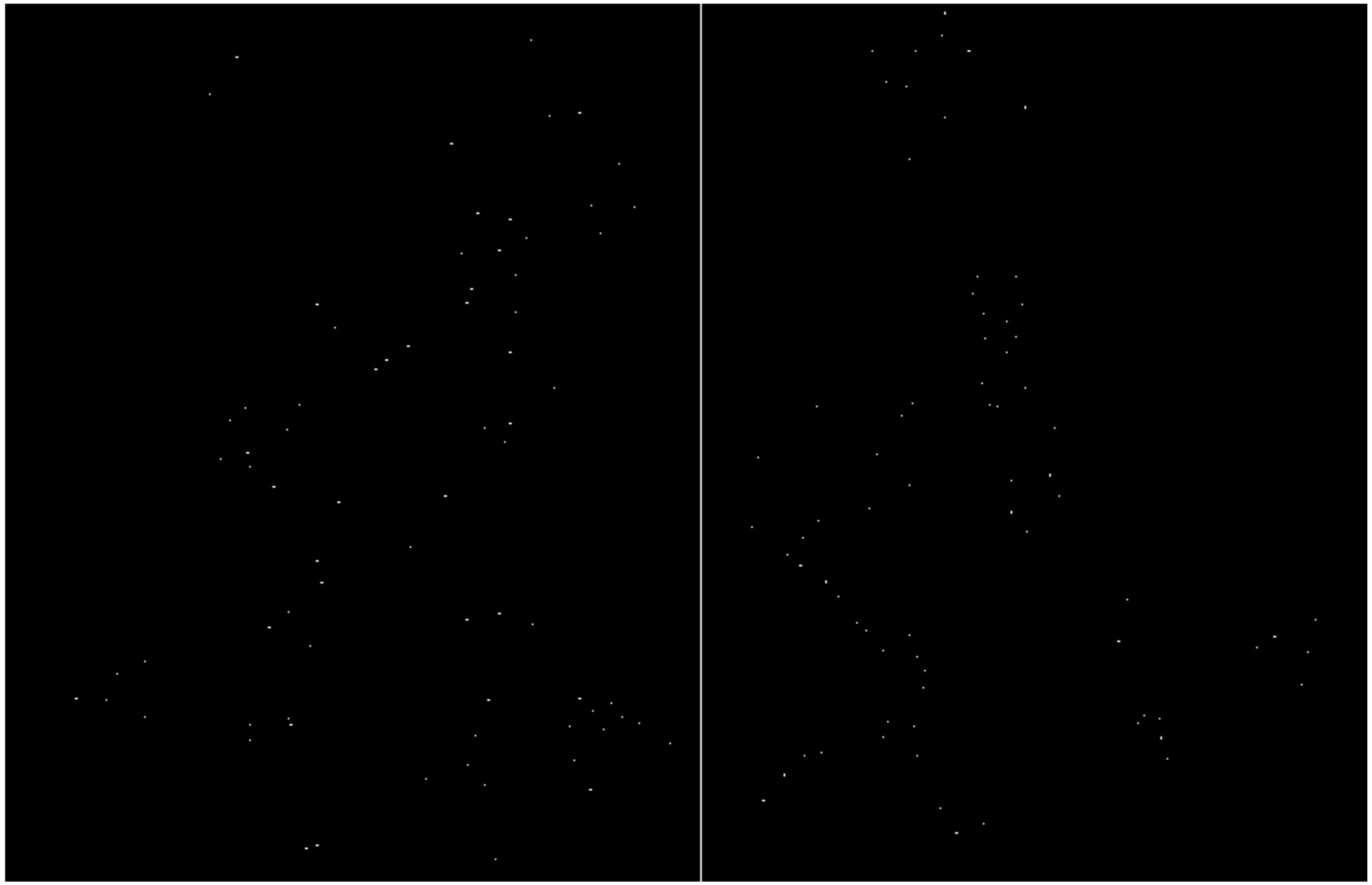
## Harris Detector: Workflow

Find points with large corner response: $R >$ threshold

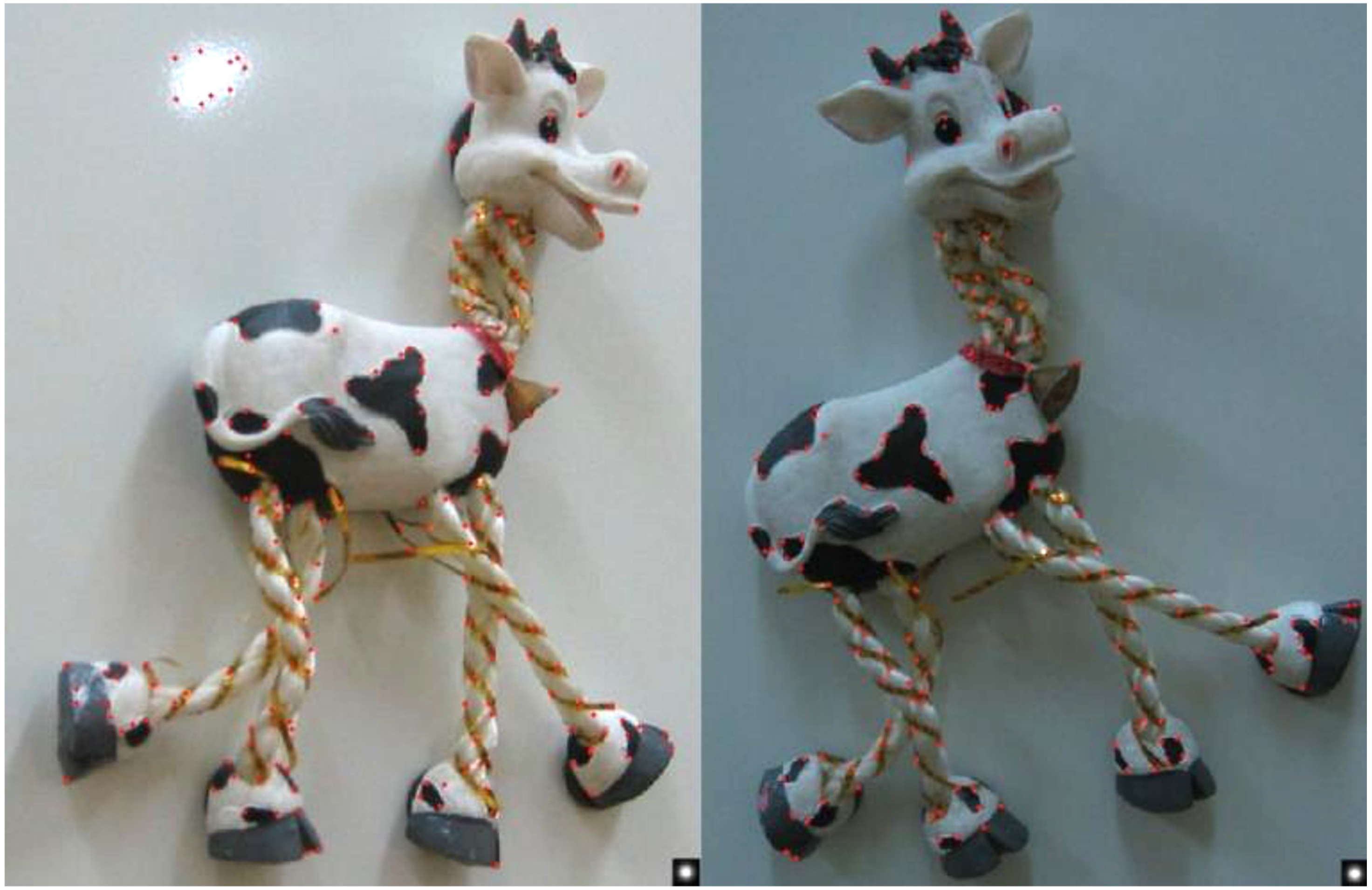


## Harris Detector: Workflow

Take only the points of local maxima of $R$



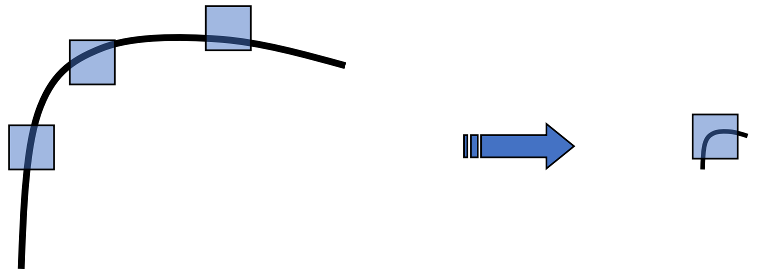
# Harris Detector: Workflow



## Harris detector, properties

The Harris and Stephens corner detector is

- Rotation invariant
- Translation invariant
- *<u>Not</u> scale invariant*



All points will be classified as edges

This is a "corner" !

## Beaudet Detector

Harris: $M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$

- The **Hessian** of a function is a matrix containing the second-order partial derivatives of the function.
- Therefore, the Hessian of the region surrounding a pixel is given by:

$$H \equiv \sum_{\mathbf{x} \in \mathcal{R}} w(\mathbf{x}) \begin{bmatrix} I_{xx}(\mathbf{x}) & I_{xy}(\mathbf{x}) \\ I_{xy}(\mathbf{x}) & I_{yy}(\mathbf{x}) \end{bmatrix}$$

- The **Beaudet detector** uses these second derivatives and is defined as the determinant of the Hessian:

$$\text{cornerness} \equiv I_{xx} I_{yy} - I_{xy}^2 \qquad (\text{Beaudet})$$

## Other feature detectors/ descriptors

- Some examples are

  - SIFT, Scale Invariant Feature Transform
  - FAST, Features from Accelerated Segment Test
  - SURF, Speeded-Up Robust Features

## SIFT feature **detector**

- SIFT- Scale Invariant Feature Transform was published by David Lowe in 1999

1. The first step is to construct a scale space
2. Thereafter approximate a Laplacian pyramid of the image using DoG using the scale space images.
3. Determine, for every pixel and for every scale, whether the pixel is a local maximum among its 26 neighbors. ( 8 in same scale, 9 in neighbor larger scale, 9 from neighbor smaller scale)

4. Discard bad key-points, i.e. untextured areas or along intensity edges. A technique similar to Harris corner detection in used here. The Hessian is found from the DoG images already calculated

   Discard keypoints with low DoG value ( low contrast)

   Discard keypoints on edges based on finding “edgeness” from the Hessian matrix

1. The first step is to construct a scale space

https://aishack.in/tutorials/sift-scale-invariant-feature-transform-scale-space/

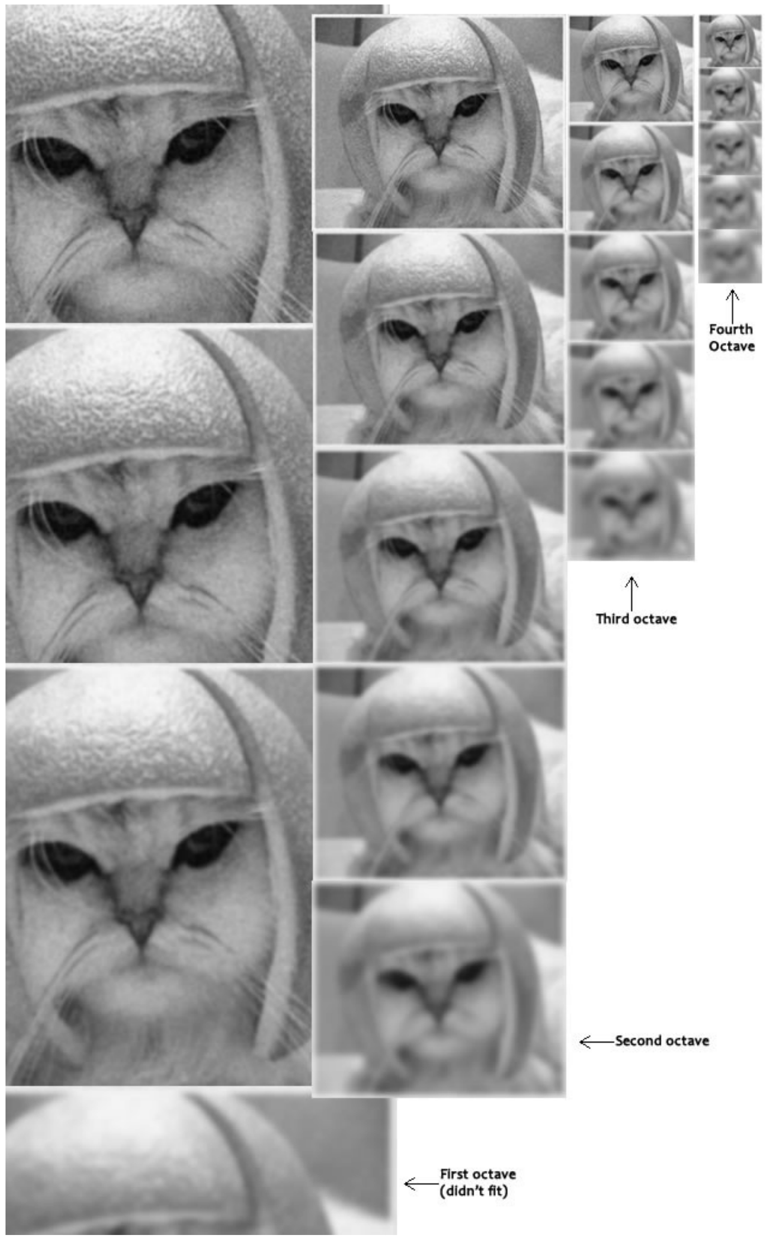
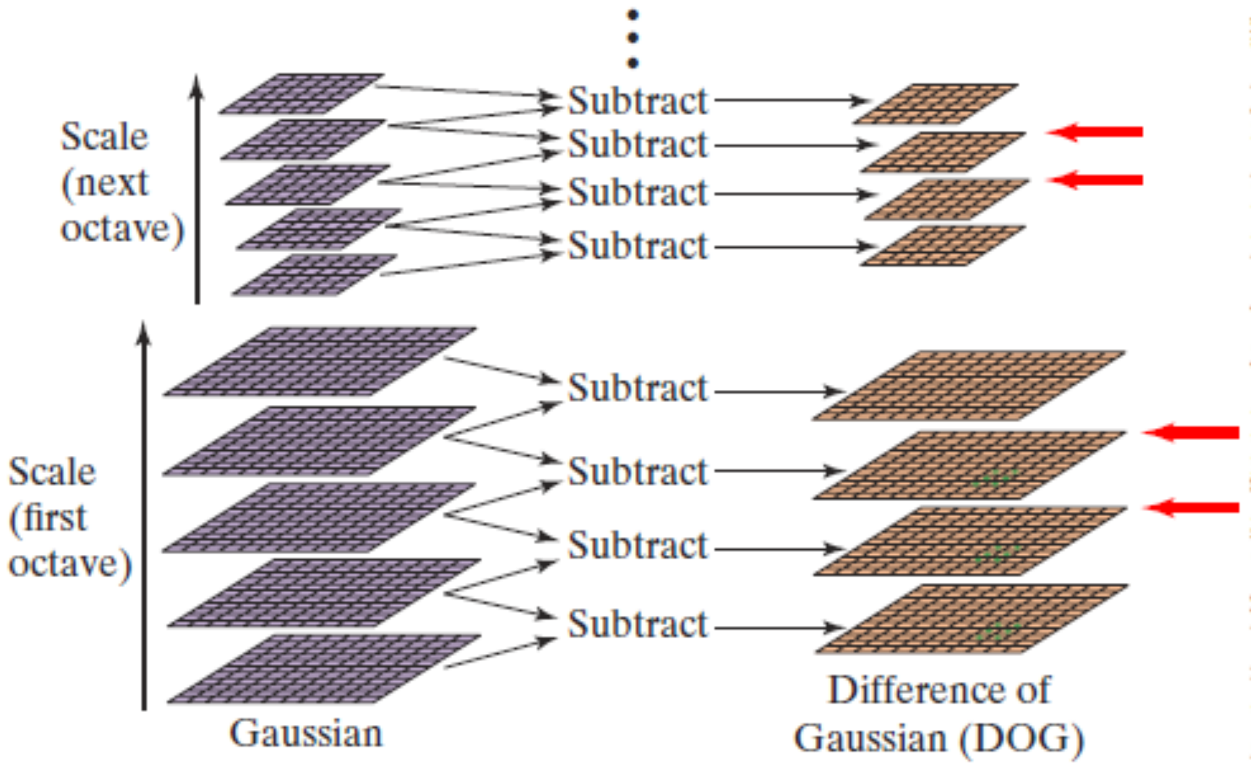


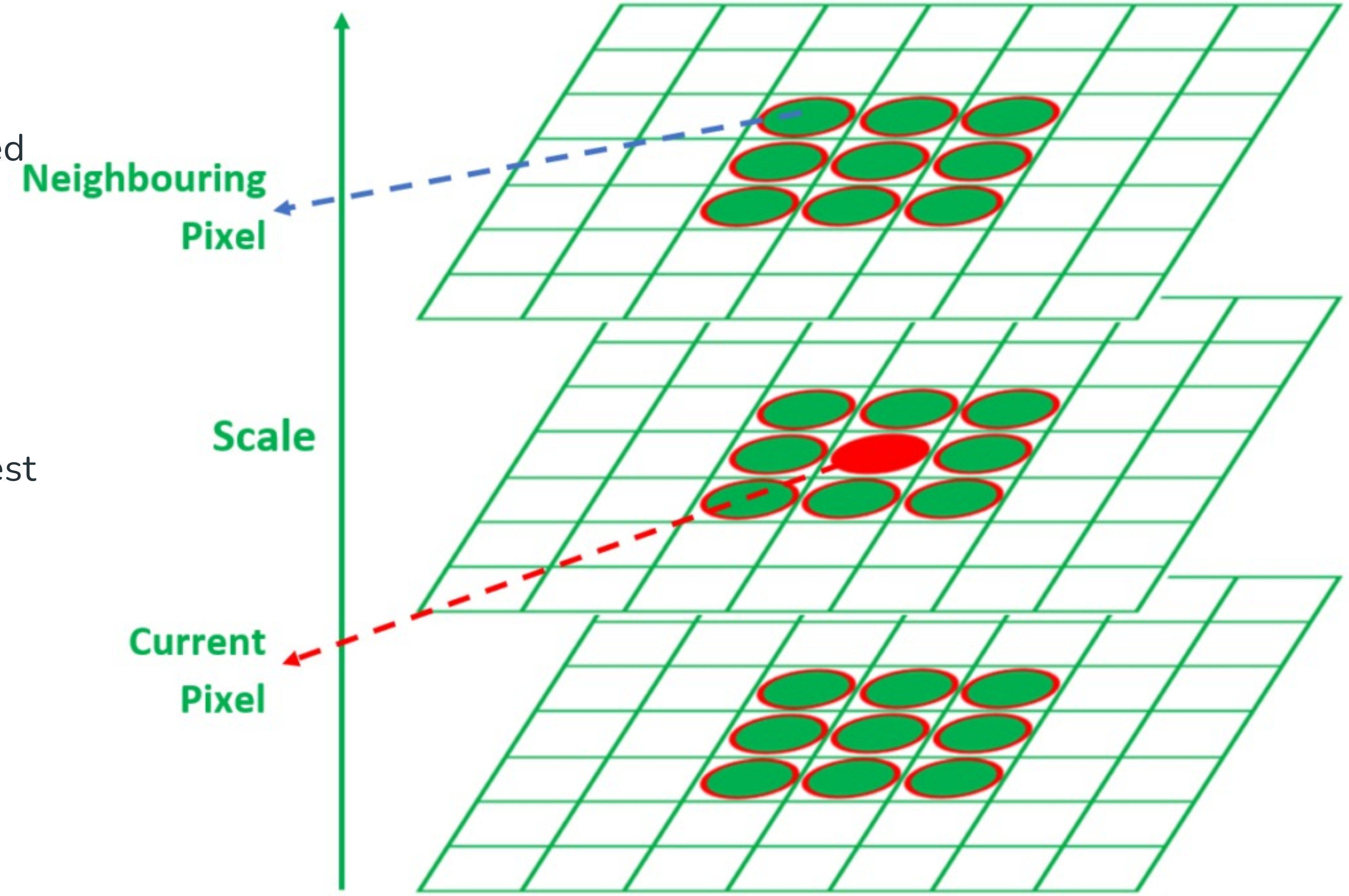
**Figure 7.17** SIFT features are detected by computing a Laplacian pyramid, then looking for local maxima among the 26 neighbors of a pixel. For each octave in this drawing there are 5 Gaussians, 4 LoGs (approximated by DoGs), and two scales (indicated by red arrows); the remaining two LoGs are used only in the neighborhood computation.

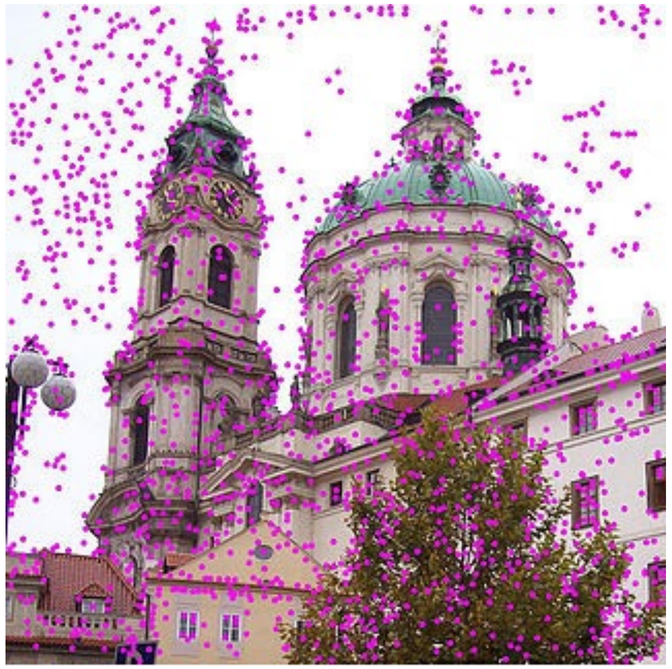
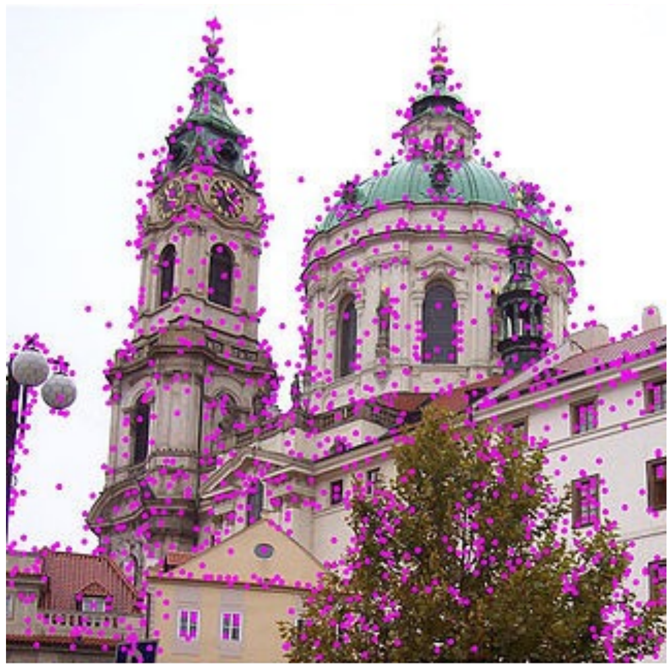
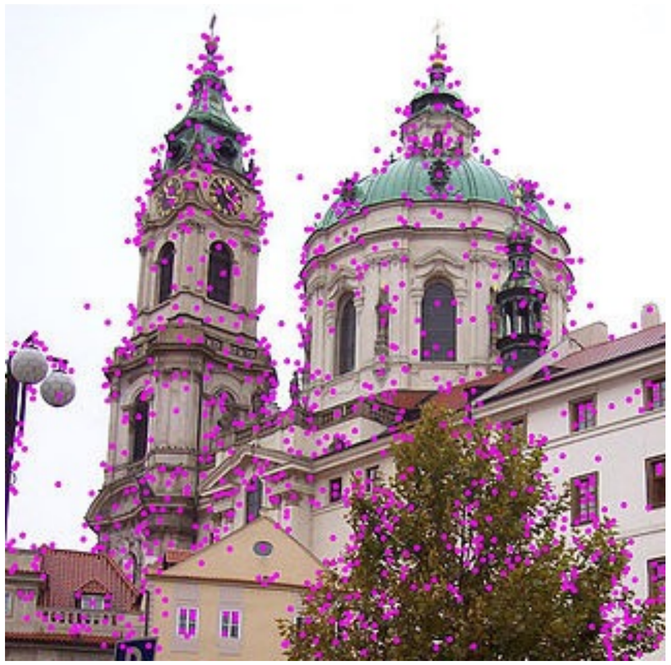


2. Thereafter approximate a Laplacian pyramid of the image using DoG using the scale space images.

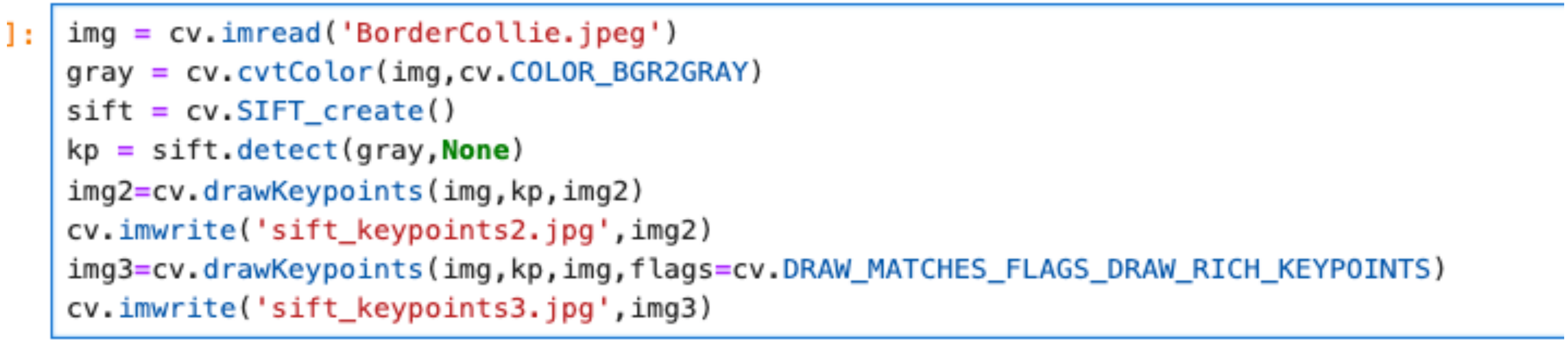
If an edge is detected at the **same location in multiple scales** (indicated by zero crossings in the scale space) **then we classify it as an actual edge.**

We can detect the Interest Points using the local maxima/minima in **Scale Space of DoG**

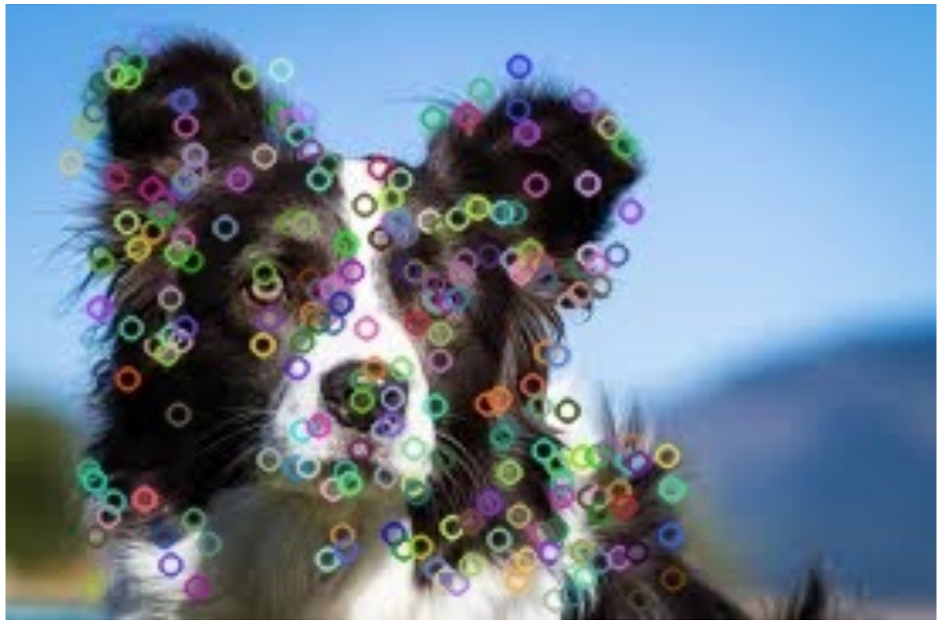


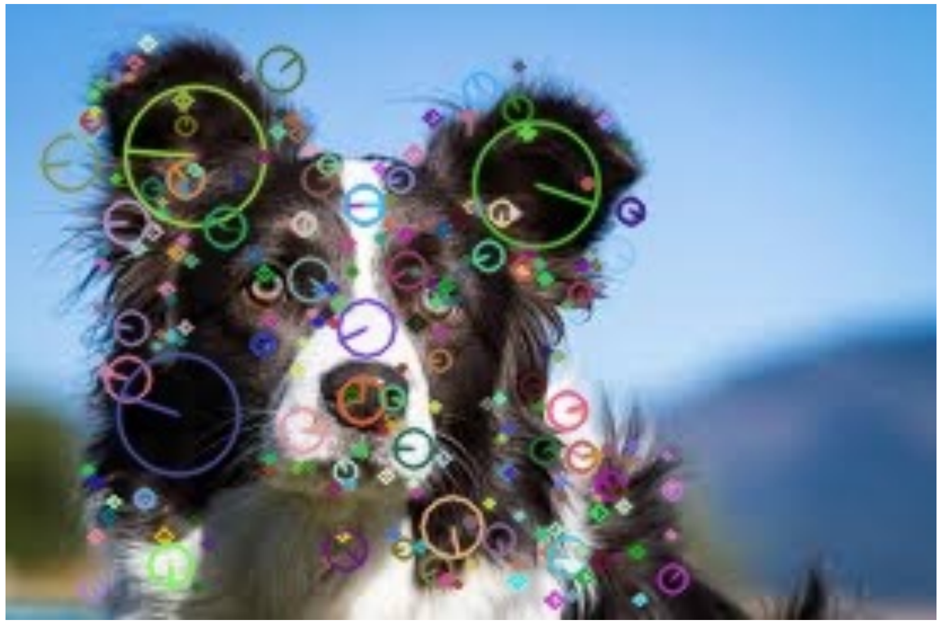




Left: After scale space extrema are detected.
Middle: the SIFT algorithm discards low-contrast keypoint.
Right:  Then keypoints located on edges are removed after looking at the edgeness using the Hessian.
Resulting set of keypoints is shown on last image.



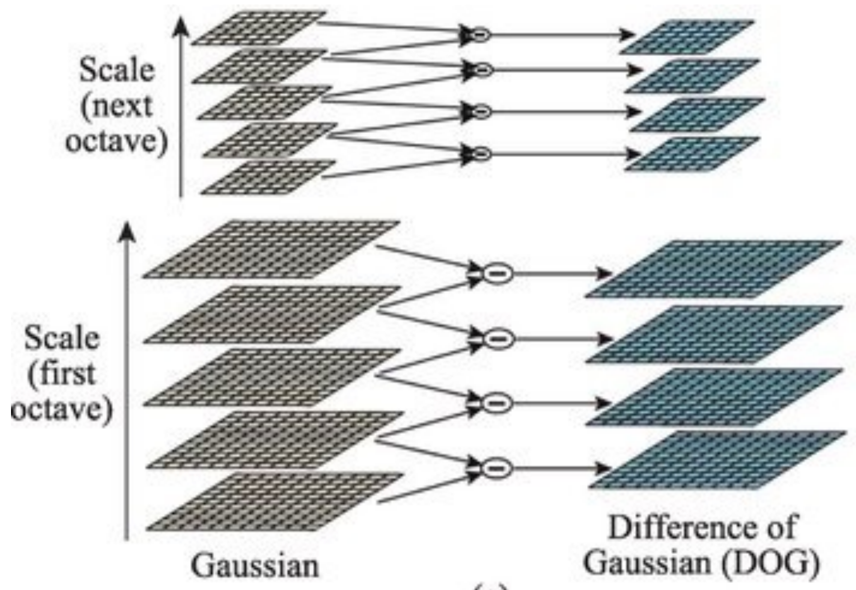
```
]: img = cv.imread('BorderCollie.jpeg')
   gray = cv.cvtColor(img,cv.COLOR_BGR2GRAY)
   sift = cv.SIFT_create()
   kp = sift.detect(gray,None)
   img2=cv.drawKeypoints(img,kp,img2)
   cv.imwrite('sift_keypoints2.jpg',img2)
   img3=cv.drawKeypoints(img,kp,img,flags=cv.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
   cv.imwrite('sift_keypoints3.jpg',img3)
```



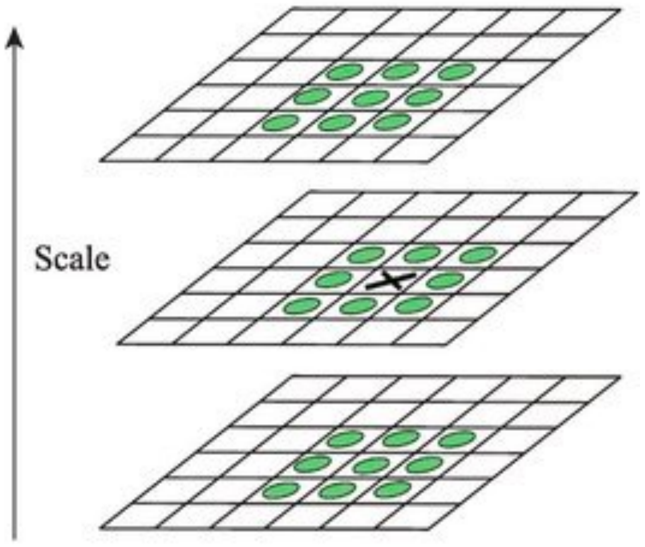


## (7.5) feature **descriptor** (SIFT)

- Around a feature point (found by the SIFT feature detector), in the same scale: Find the image gradient magnitudes and orientation in the neighborhood. Scale decides size of neighborhood and gradient smoothing.
- Find dominant gradient orientation, make all orientations relative to that. (to make it rotational invariant)
- Weight by a Gaussian to give more emphasize to the orientation close to the feature point (center)
- Make into 3D histogram over position and orientation
- Values in histogram are concatenated to form a vector that describes the feature point.


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

(a)


Scale

(b)

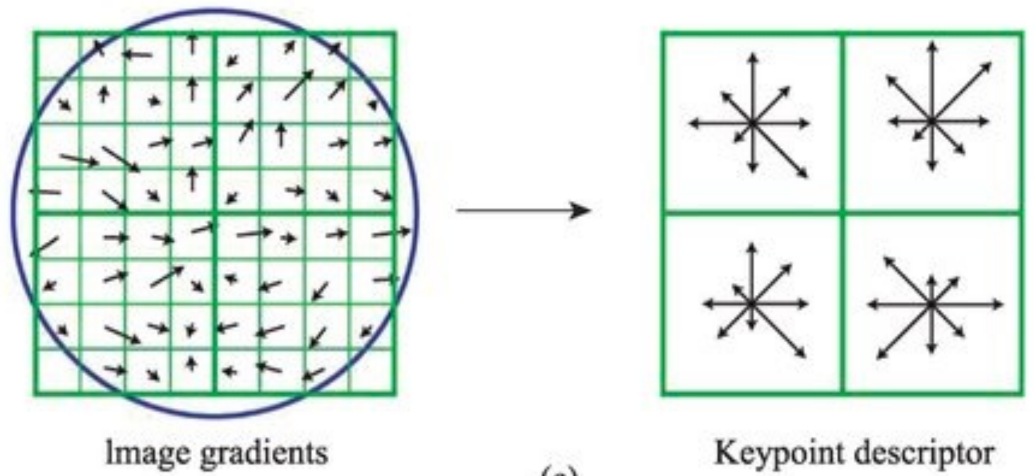


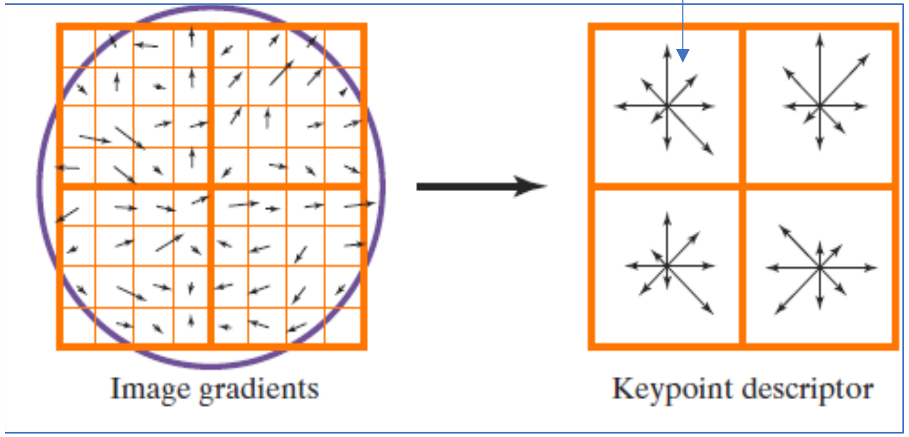
Image gradients
Keypoint descriptor

(c)

To find the descriptor:  considering the neighborhood of the keypoint and calculate the magnitude and direction of gradients  ( remember sobel..) of the neighborhood.  Accumulate over the neighbourhood and over different scales.

All gradient orientation is relative to the dominant orientation  -> invariant to rotation.

8 directions.  Here 2x2 grid, more common 4x4 grid. This gives 8x4x4 = 128 values in a feature vector. Vector is normalized – less dependent on illumination variance



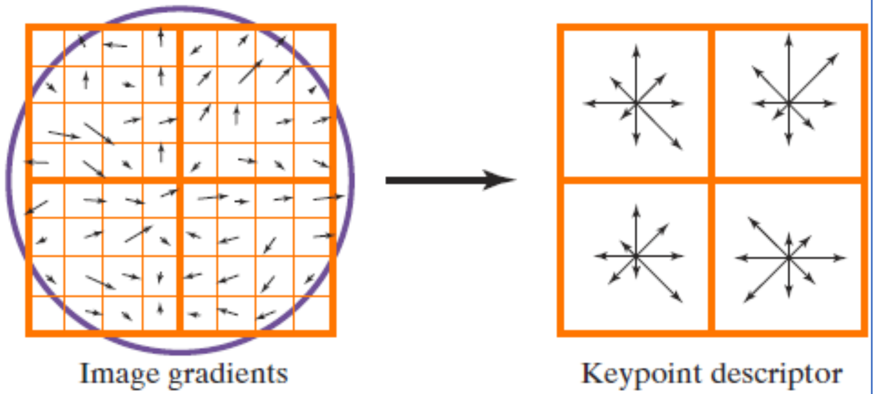
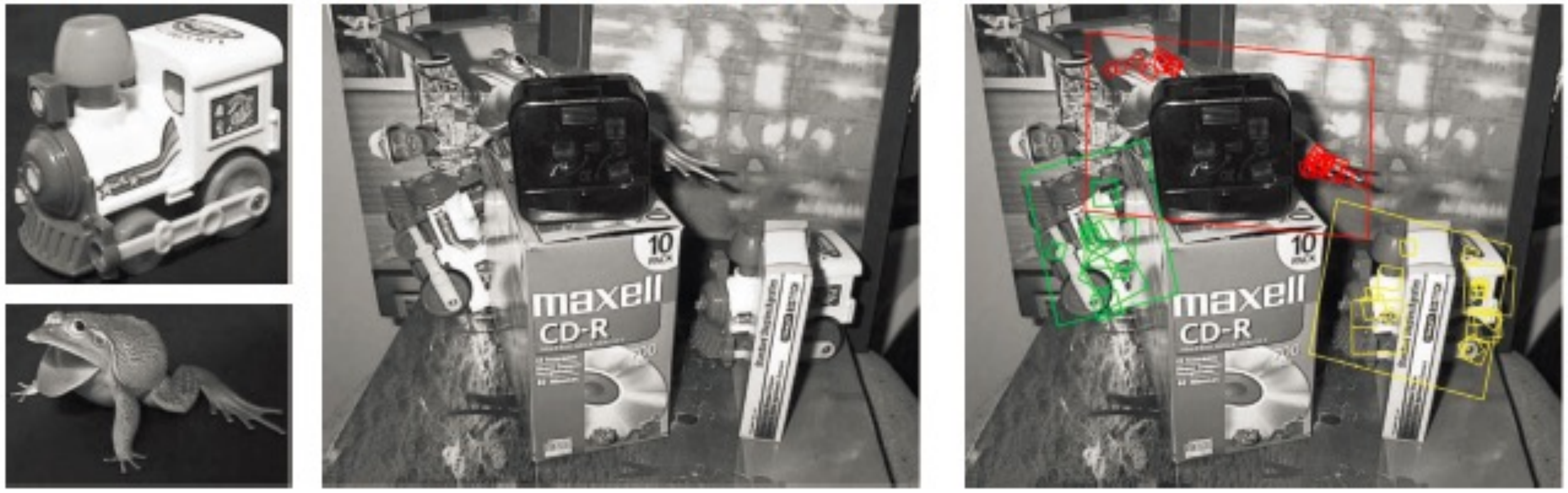


**Figure 7.18** The SIFT feature descriptor is computed by accumulating the orientations of the gradient vectors in a neighborhood of the feature point into a 3D array over position and orientation.

**Figure 7.19** SIFT feature matching results. SIFT feature descriptors from the query image (middle) are matched against descriptors from the database (left) to detect objects at various poses and lighting conditions, and even with severe occlusion (right).

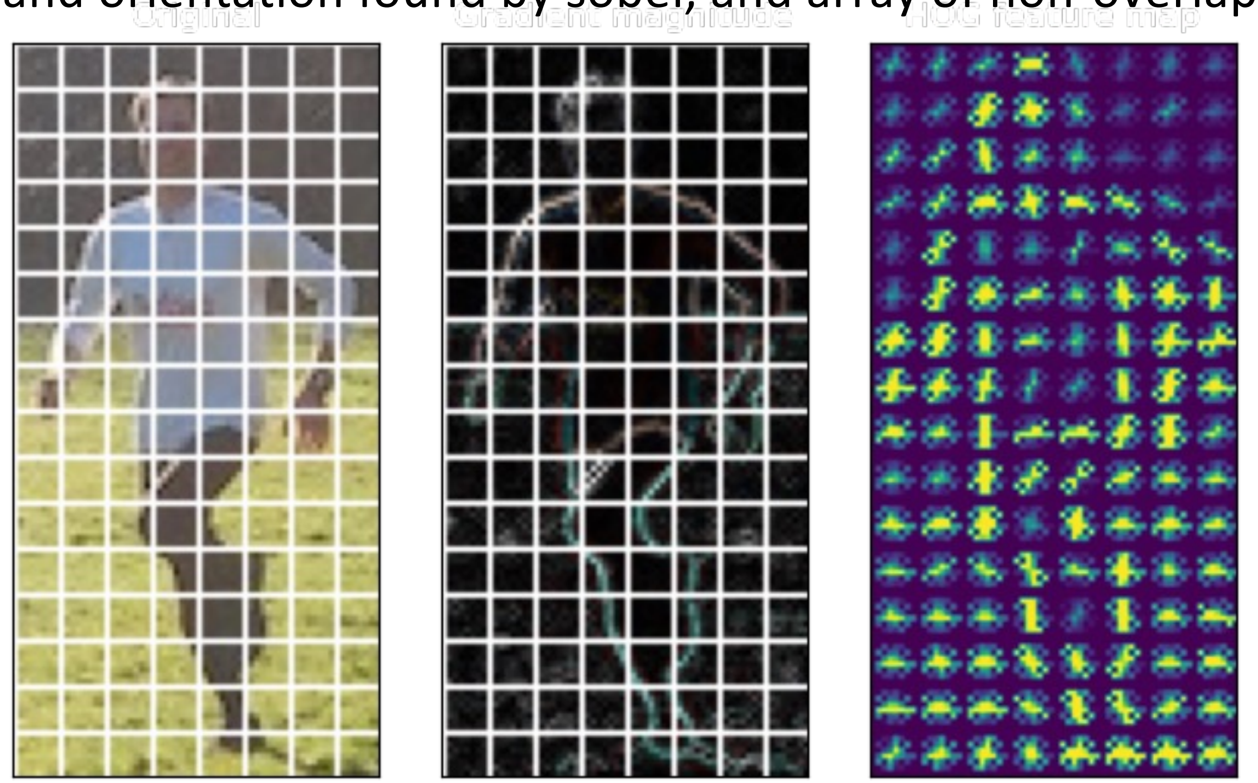


## Other feature descriptors

- GLOH – Gradient location and orientation Histogram
  - Extension of SIFT, slightly better than SIFT in some situations
- SURF – speeded up robust features
  - Much faster than SIFT , using square filters instead of gaussian etc. Both feature detector and descriptor.

HOG – Histogram of oriented Gradients, usually meant as calcualting the descriptor on all points in the image, not just corner/feature points.

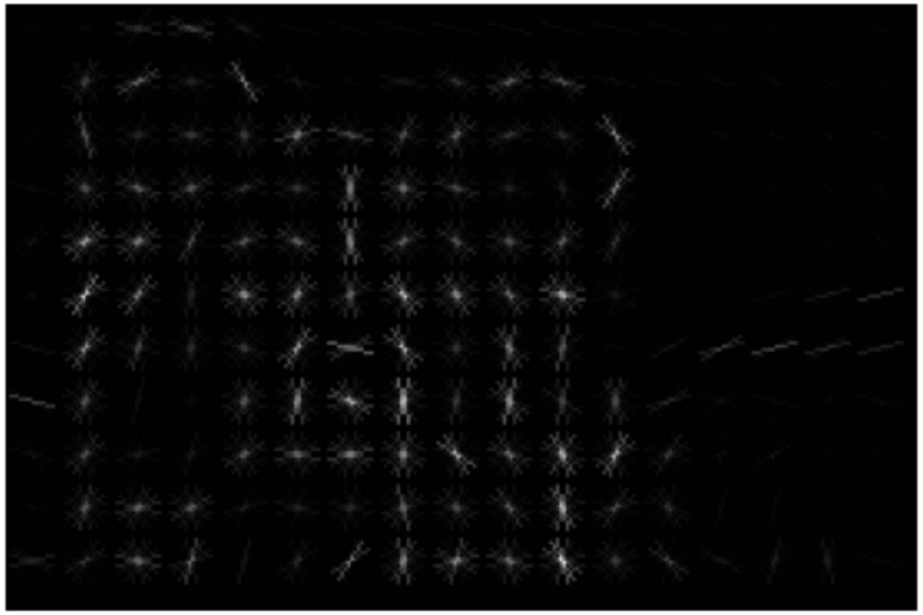
Gradient and orientation found by sobel, and array of non-overlapping 8x8  cells



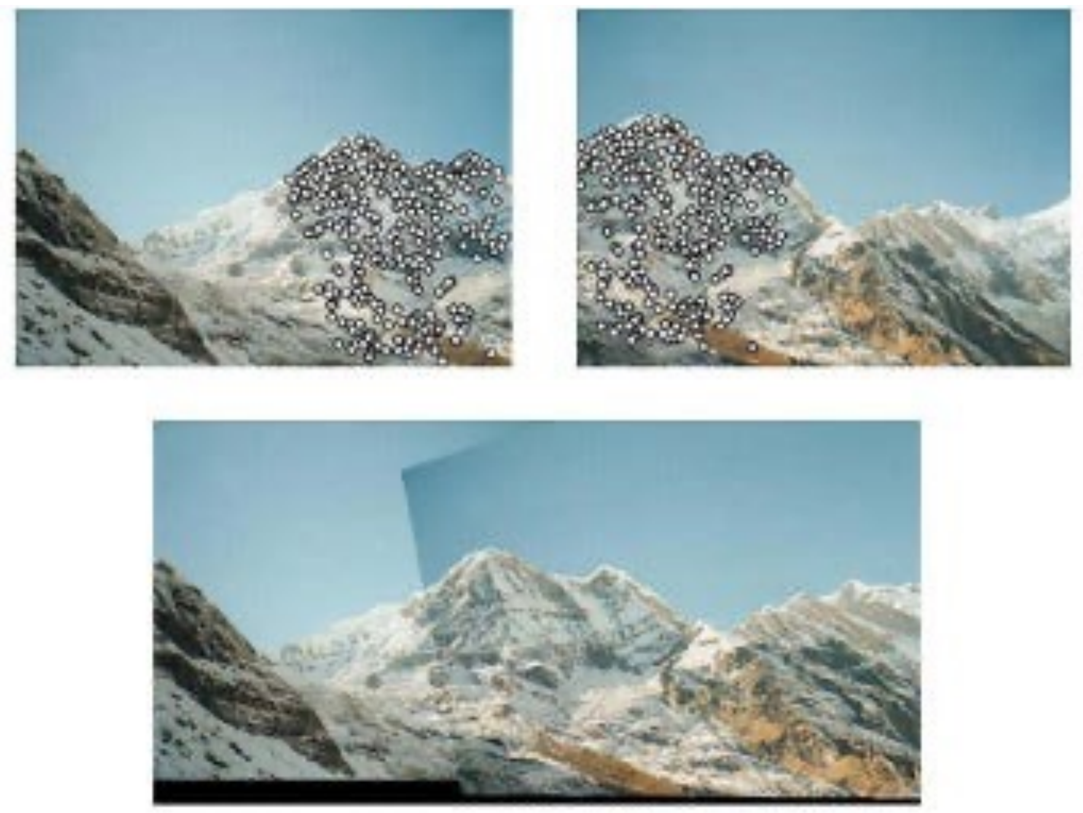
https://www.neuralception.com/featureextractors-2/

Input Image

HOG Features



## Panorama stitching – example application



•**Step #1:** Detect keypoints (DoG, Harris, etc.) and extract local invariant descriptors (SIFT, SURF, etc.) from the two input images.

•**Step #2:** Match the descriptors between the two images.

•**Step #3:** Use the RANSAC alg. to estimate a homgraphy matrix using our matched feature vectors.

•**Step #4:** Apply a warping transformation using the homography matrix obtained from **Step #3**.

## Lines, corners and feature points in images

- Three  points from the topic:

1. What are important (corner) points in an image?
   - ✓ Where there is large changes if silding a litle box in any directions
2. How can we make features from corner-points?
   - ✓ Finding gradient magnitudes and orientations(sobel) in neighbourhood, converting into feature vector
3. Some applications that use corner-points and features
   - ✓ Stitching images together (panorama ), finding objects, describing images or objects, finding humans