EE 604
Digital Image Processing

IIT KANPUR
Indian Institute of Technology, Kanpur

Tanaya Guha

Aug - Nov 2017

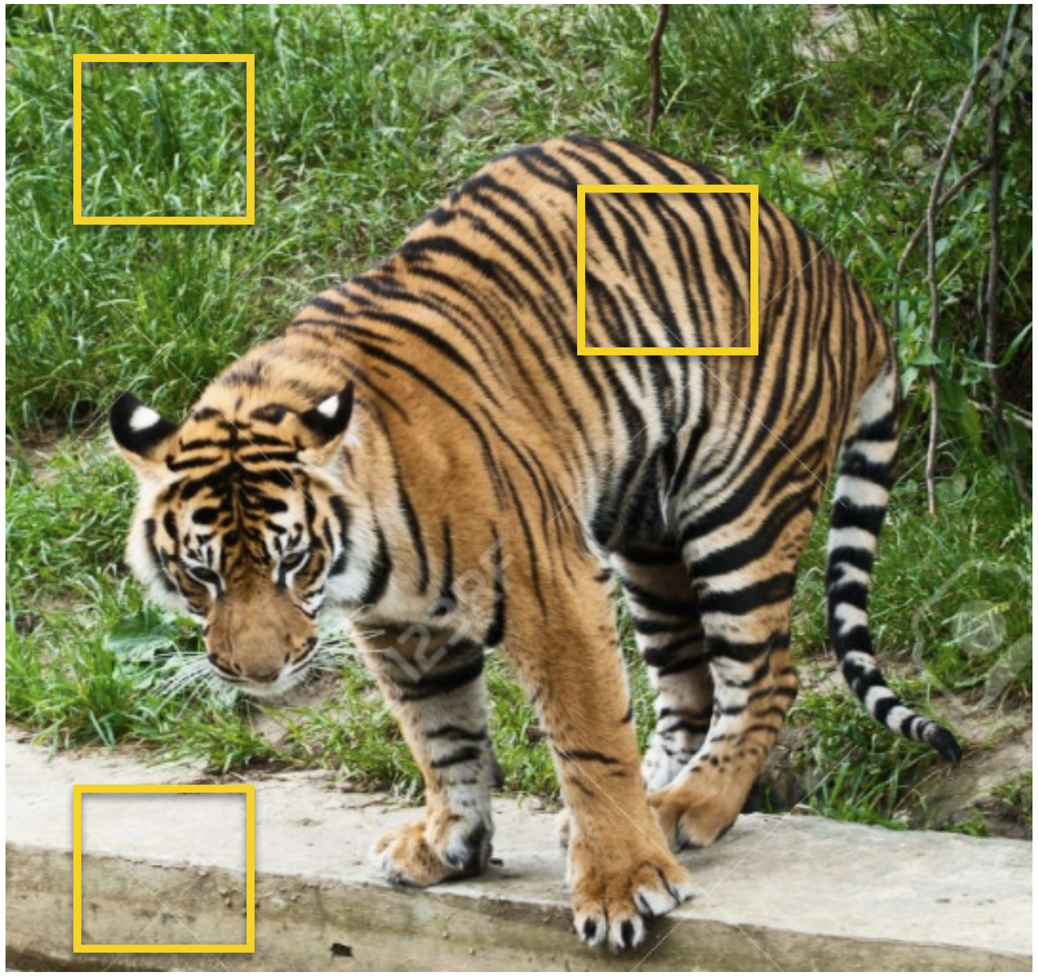
Image representation

- **Objective:** Capture the properties of an image suitable for matching, recognition and analysis.
- **Types of representation**
  - Global vs. Local
  - Captures specific image property
    - shape, color, texture, etc.
    - motion, change, deformation

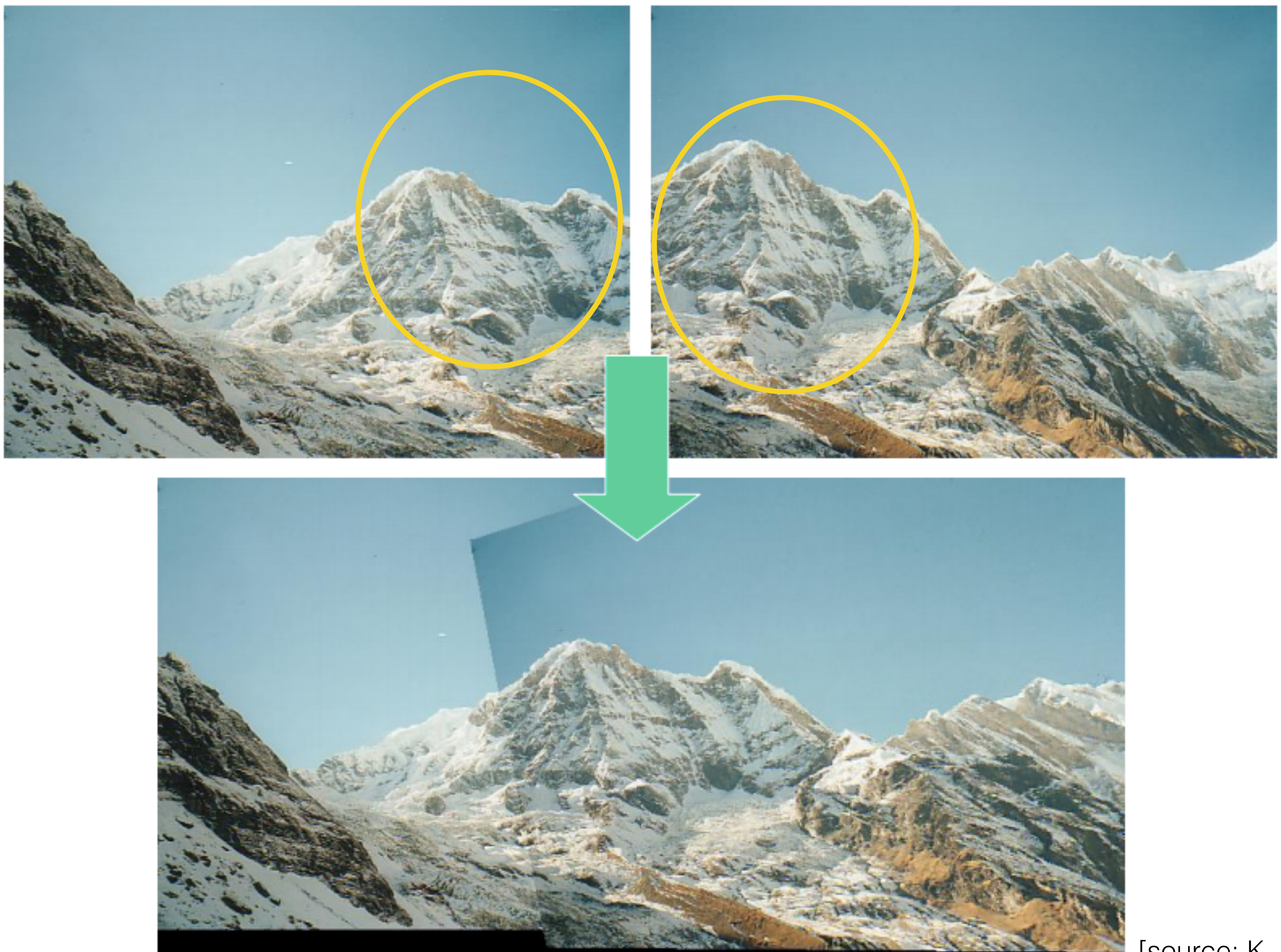
Image representation

Properties of a good image descriptor

- Robust against noise
- Compact representation
- Invariant to geometric distortions
    - Rotation, translation
- Invariant to scale
- Invariant to photometric distortions
    - Illumination, contrast

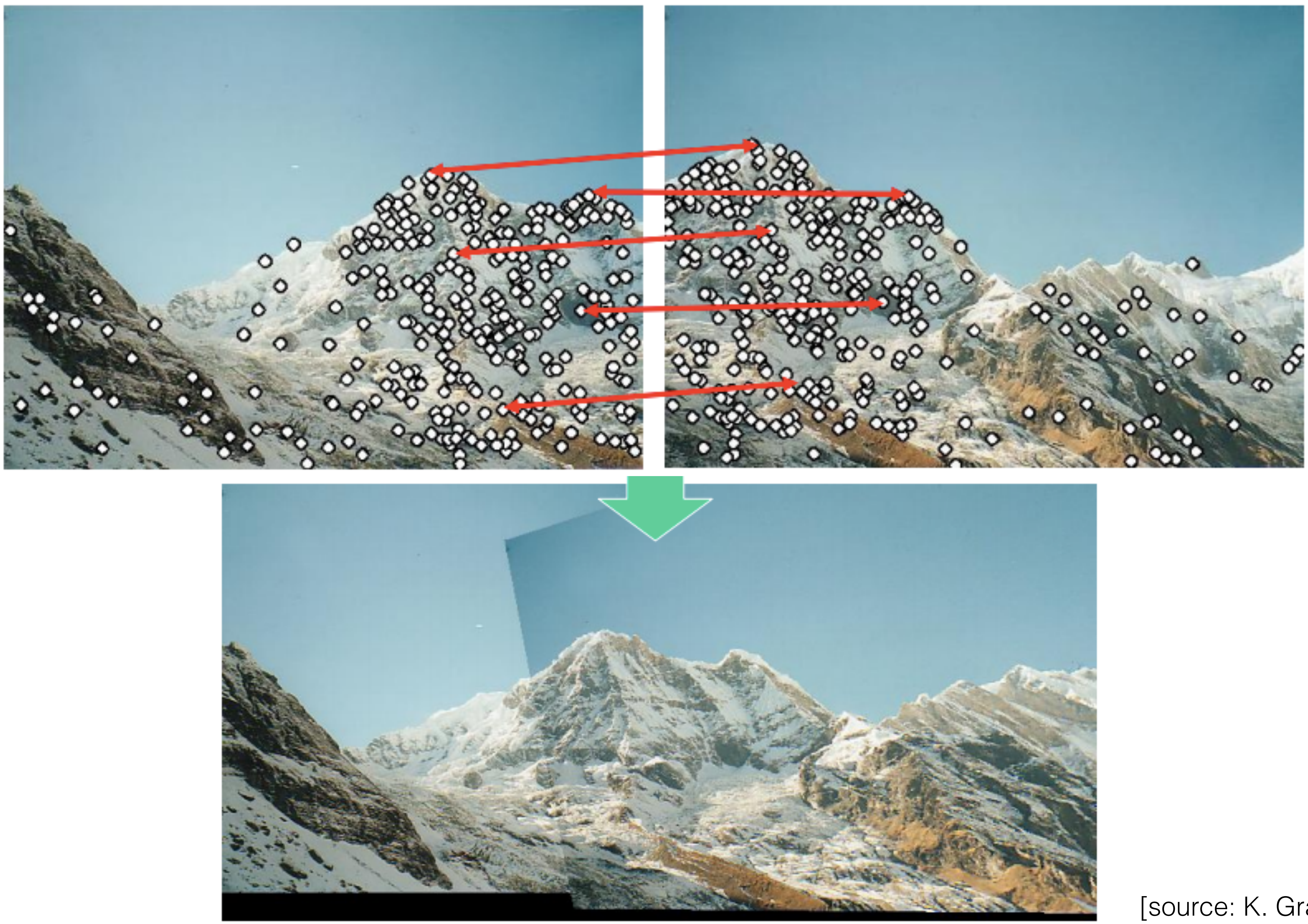


Panorama creation



[source: K. Grauman]

Image representation



[source: K. Grauman]

Image matching

[source: K. Grauman]

Global vs. Local features

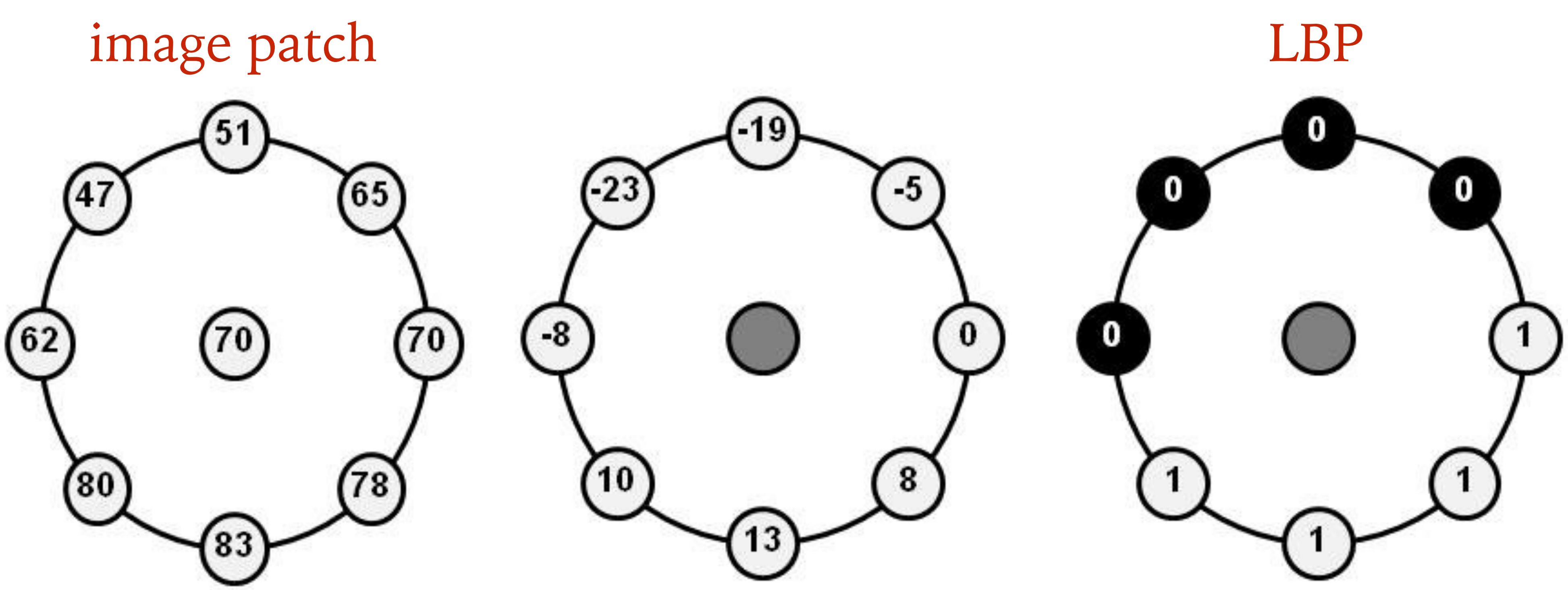
- **Global features**
  - Histogram, Fourier coefficients,…
  - Problems with global features?
- **Local** features are often preferred over global ones
    - Robust to occlusion and clutter
    - Large number of such features can be extracted from a single image
    - Can better distinguish objects.

Image features

- **Two important aspects of feature-based representations**
    - How to encode texture, shape, color information in an image?
    - Which regions in the image to encode?

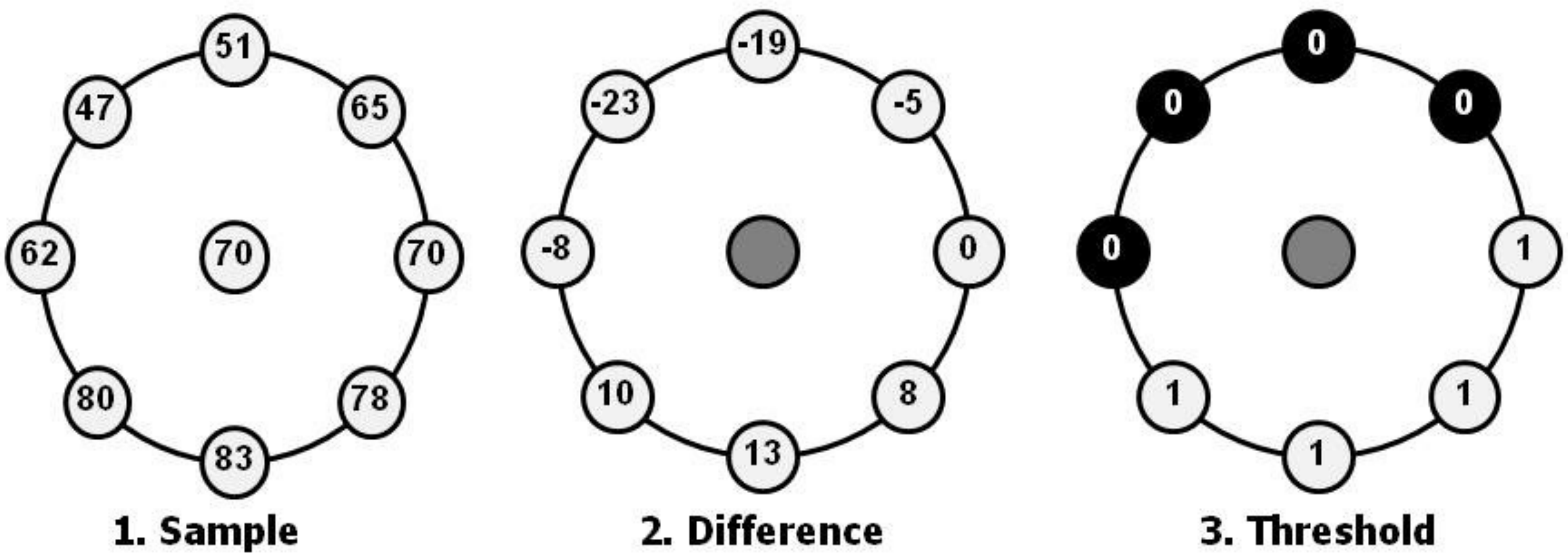
Local Binary Pattern

- **LBP** was introduced by Ojala et al. in 1990 as a texture feature.
- Textures
    - have no strict definition
    - characterized by complex patterns composed of sub-patterns
    - gives rise to perceived lightness, uniformity, smoothness, granularity



[source: Ojala et al.]

Local Binary Pattern



1*1 + 1*2 + 1*4 + 1*8 + 0*16 + 0*32 + 0*64 + 0*128 = 15

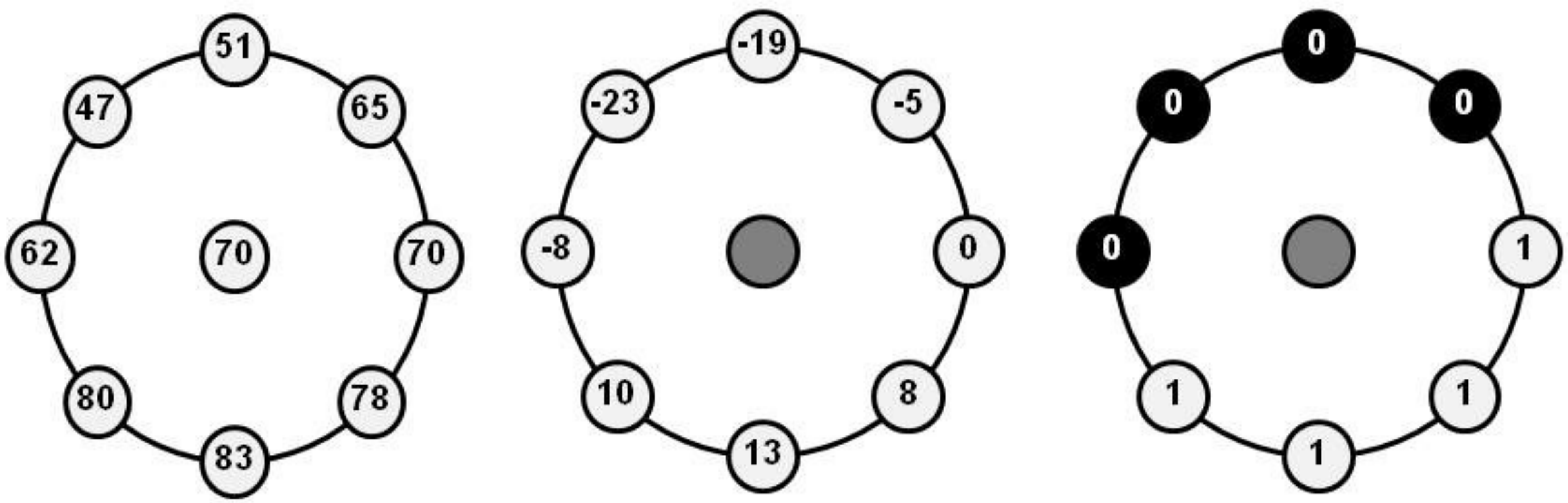
LBP code

[source: Ojala et al.]

Local Binary Pattern

The value of the LBP code of a pixel (x_c, y_c) is given by:

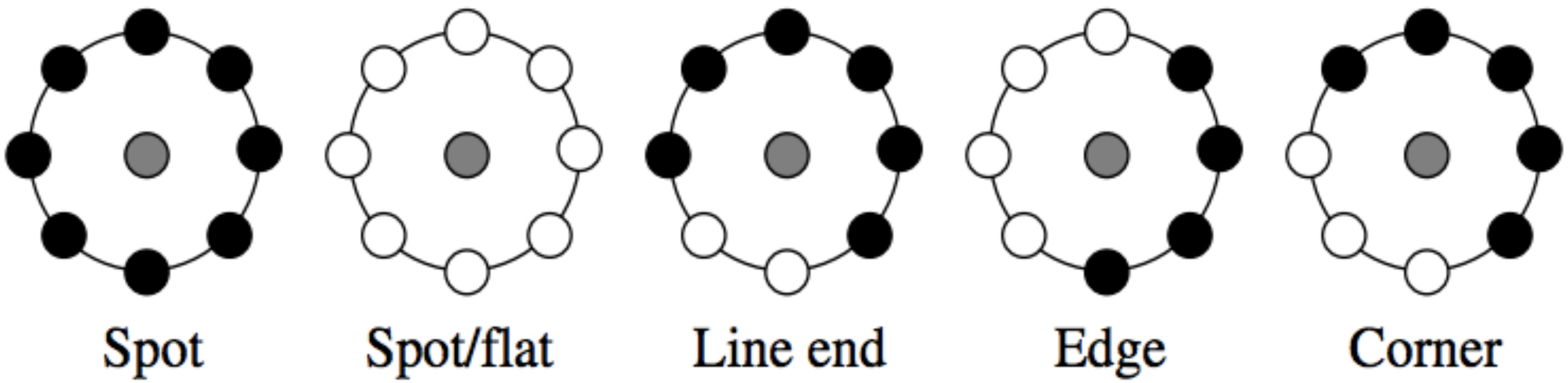
$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p \qquad s(x) = \begin{cases} 1, & if\ x \geq 0; \\ 0, & otherwise. \end{cases}$$

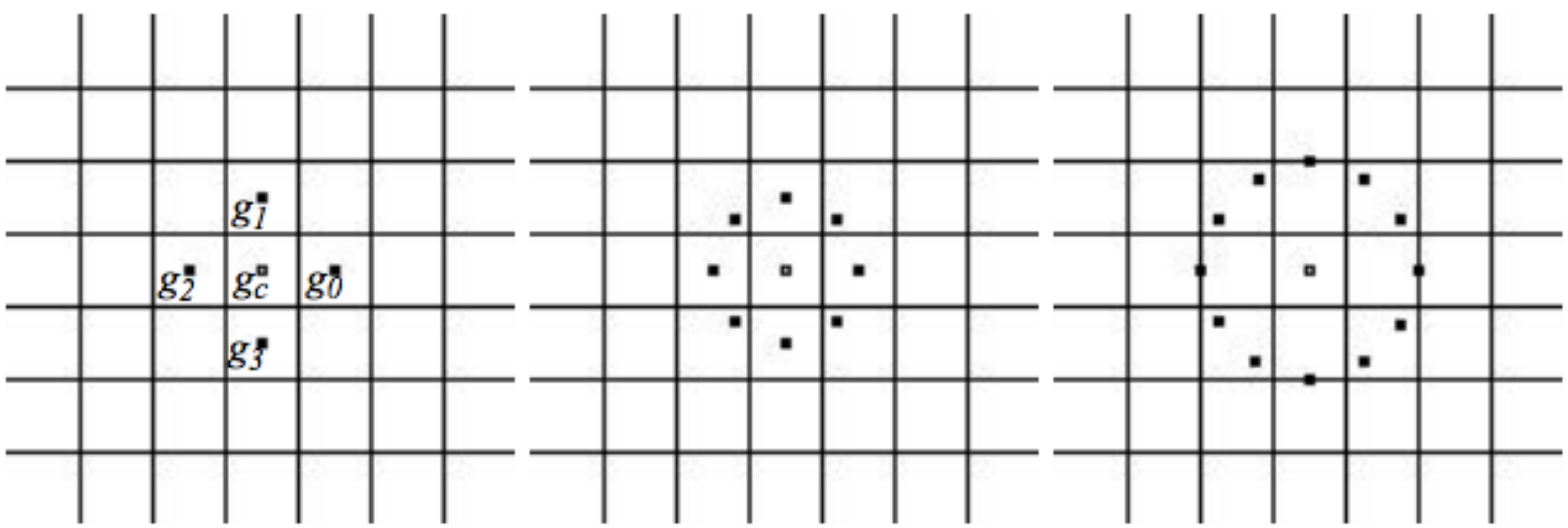


1. Sample   2. Difference   3. Threshold

[source: Ojala et al.]

Local Binary Pattern





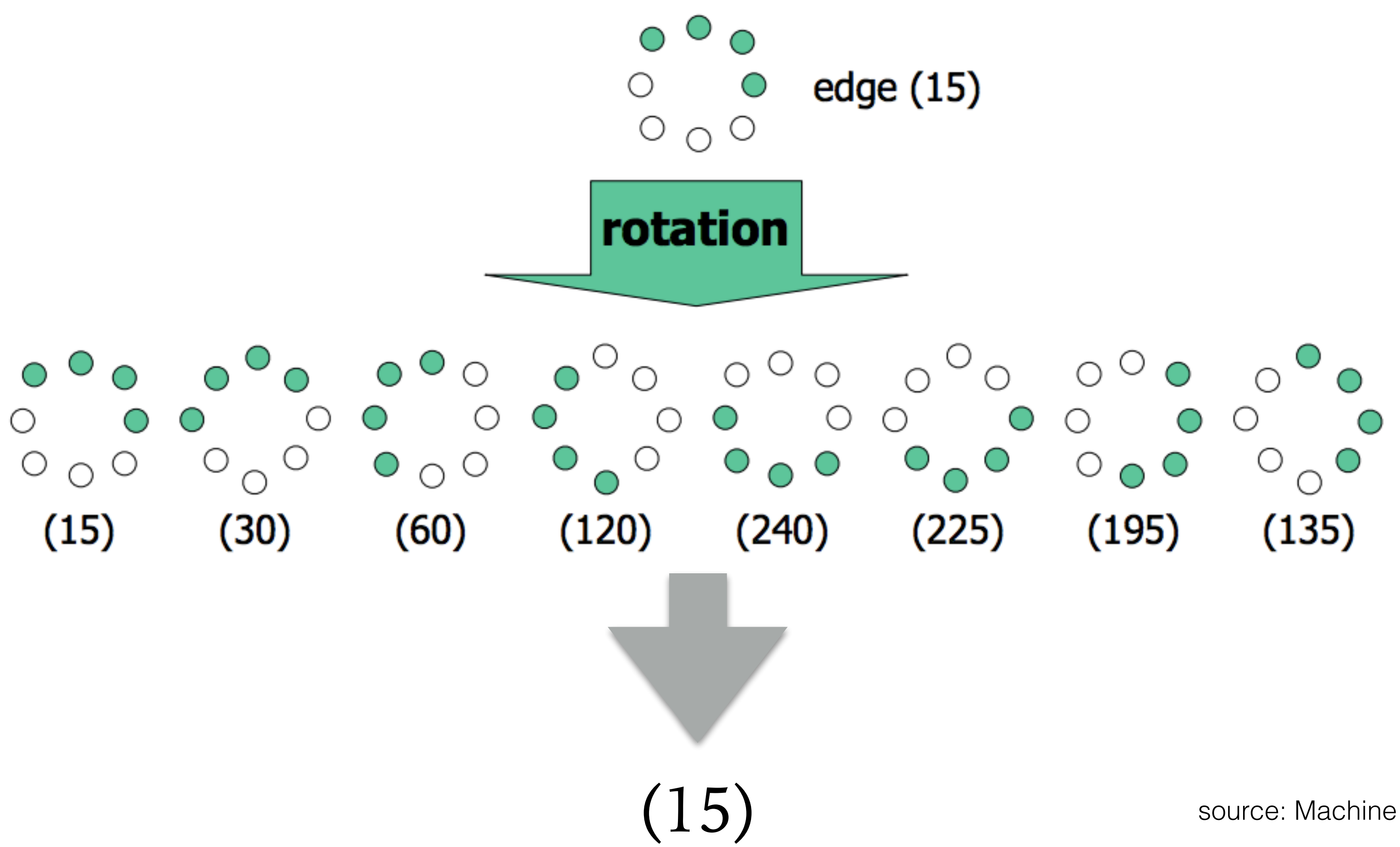
Local Binary Pattern

- If using 8 neighborhood points, there are 256 possible LBP codes.
- Each pixel is represented by one of these 256 LBP code.
- Compute a 256 dimensional LBP histogram.
    - This indicates the probability of occurrence of each LBP code in an image.
- A global LBP histogram can be created for the entire image.
- Or, an image can be divided into sub-regions, and then one LBP histogram for each sub-region can be computed.
    - How would this help?

source: Machine vision group, U of Oulu

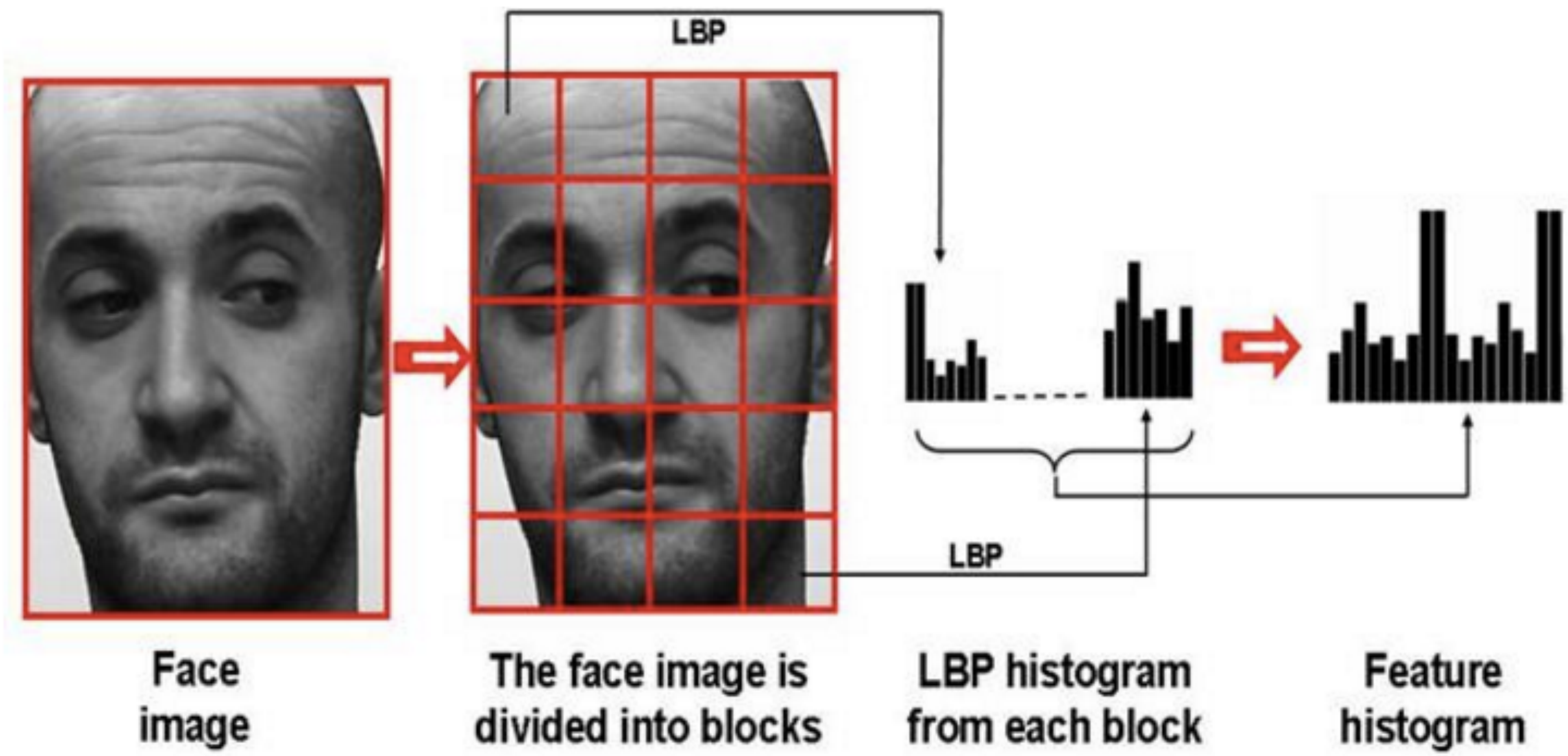
Local Binary Pattern

- Rotation will change the binary pattern and the LBP code
- Can we make LBP code rotation invariant?



source: Machine vision group, U of Oulu

Facial LBP features

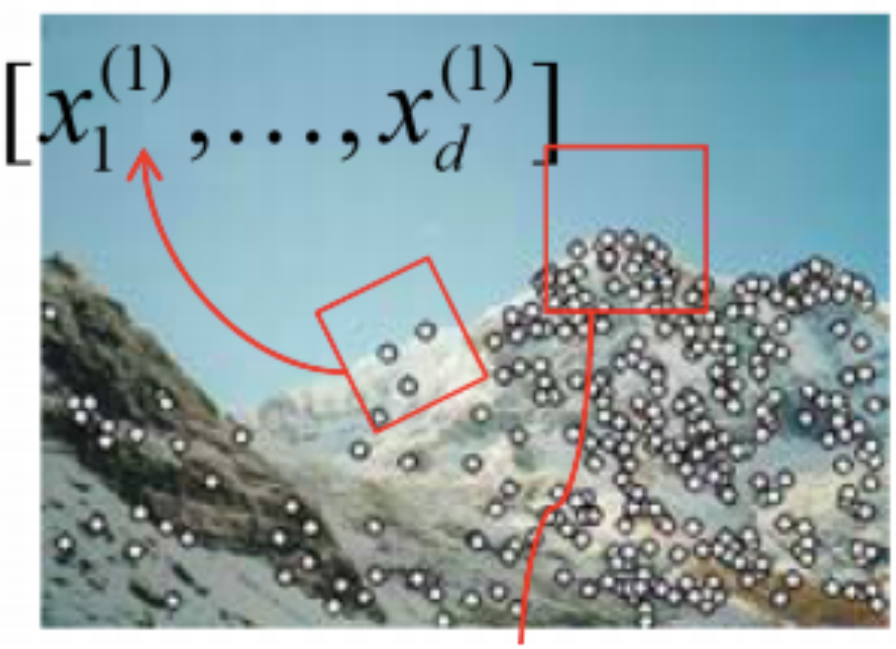


[source: Machine vision group, U of Oulu]

Image features

- Some local image features extract information from the entire image (dense)
- Many other features first identify some **interest points,** and extract information around them.

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

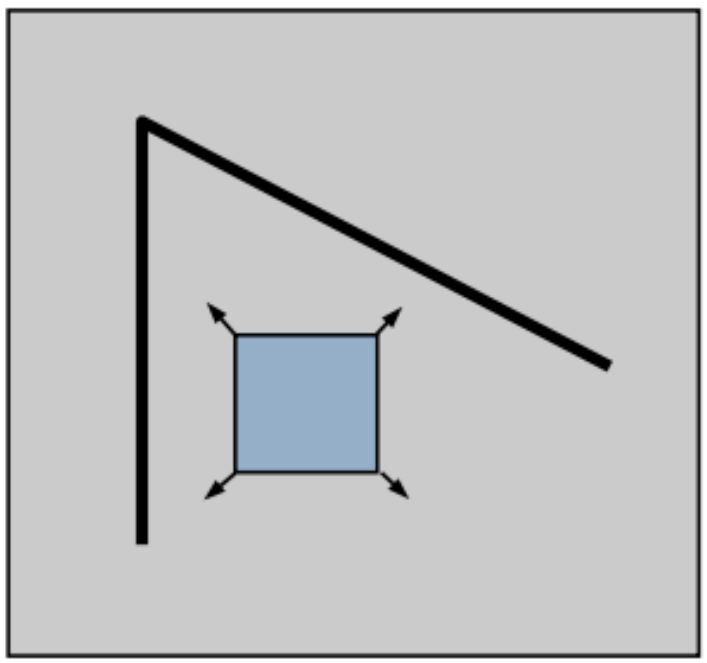


[source: K. Grauman]

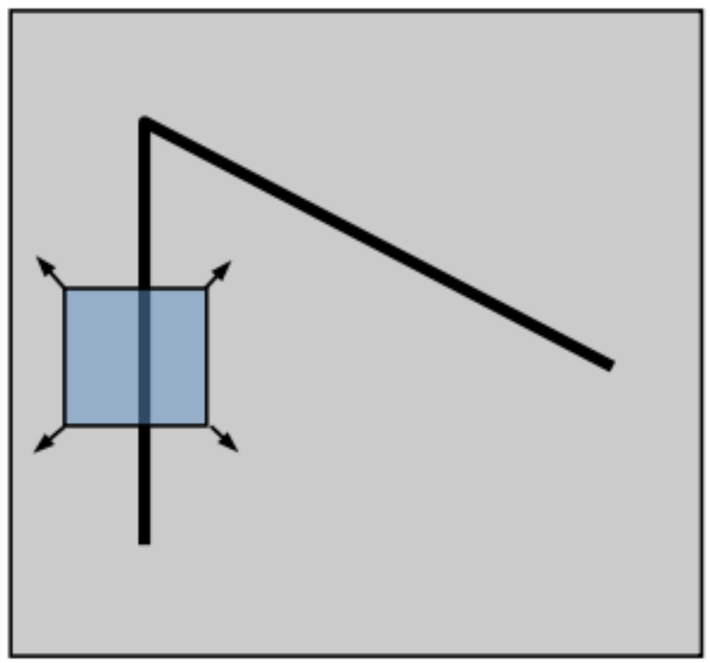
Interest points

- Good **interest points** are
    - distinctive
    - robust to transformation
    - robust to noise and distortions
    - of good quantity
- Good **interest points** are found at
    - the intersection of two edges
    - line endings
    - points on a curve or isolated points that are local extrema

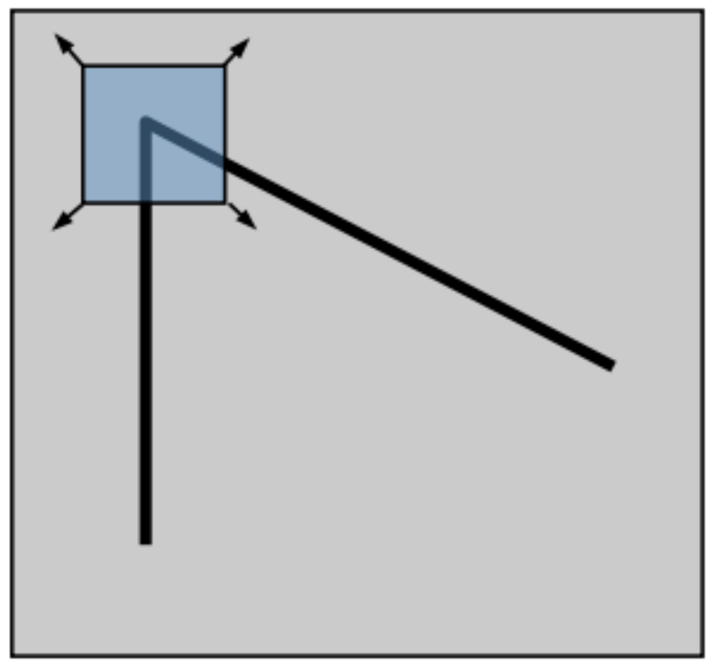
Harris corner detector



“flat” region:
no change in all directions



“edge”:
no change along the edge direction



“corner”:
significant change in all directions

[source: R. Utrausun]

Harris corner detector

Error between two patches shifted by (u,v)

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

window function

intensity at the
shifted location

- Flat regions —> small ***E(u,v)***
- Dissimilar regions —> large ***E(u,v)***
- We are looking for patches for which we get large ***E(u,v)***
- We wish to maximize $\sum [I(x+u,y+v) - I(x,y)]^2$

Harris corner detector

- Taylor series approximation

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

- Using the approximation in our original equation:

$$\begin{aligned} E(u,v) &\sim \sum_{x,y} [(I(x,y) + uI_x + vI_y)) - I(x,y)]^2 \\ &= \sum_{x,y} u^2 I_x^2 + 2uvI_xI_y + v^2 I_y^2 \\ &= \begin{bmatrix} u & v \end{bmatrix} (\sum \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}) \begin{bmatrix} u \\ v \end{bmatrix} \end{aligned}$$

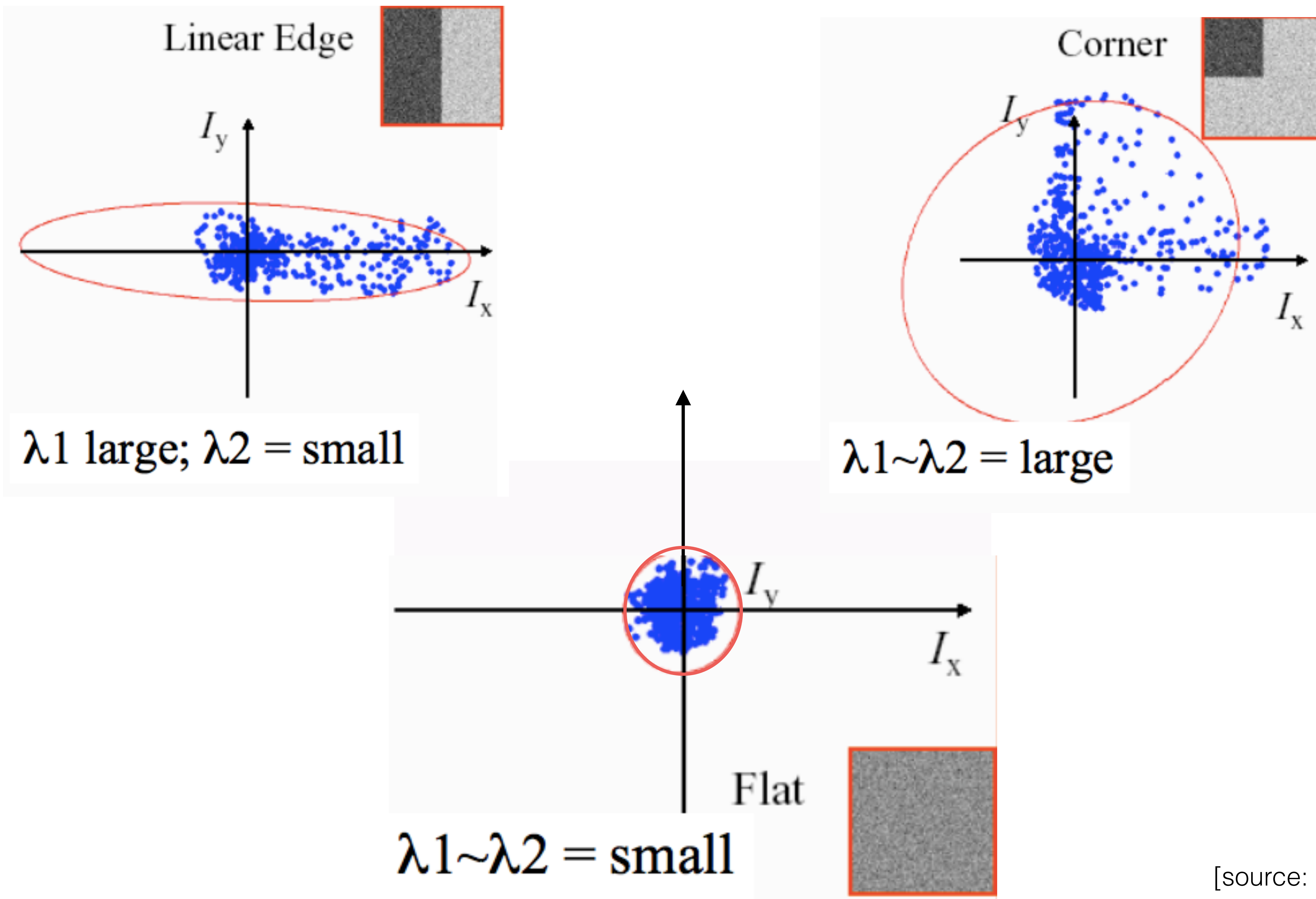
Harris corner detector

- Rewrite error as $E(u,v) = \begin{bmatrix} u & v \end{bmatrix} \mathbf{M} \begin{bmatrix} u \\ v \end{bmatrix}$

  where,

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

Harris corner detector



[source: R. Collins]

Harris corner detector

- Rewrite error as $E(u,v) = \begin{bmatrix} u & v \end{bmatrix} \mathbf{M} \begin{bmatrix} u \\ v \end{bmatrix}$

where,

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

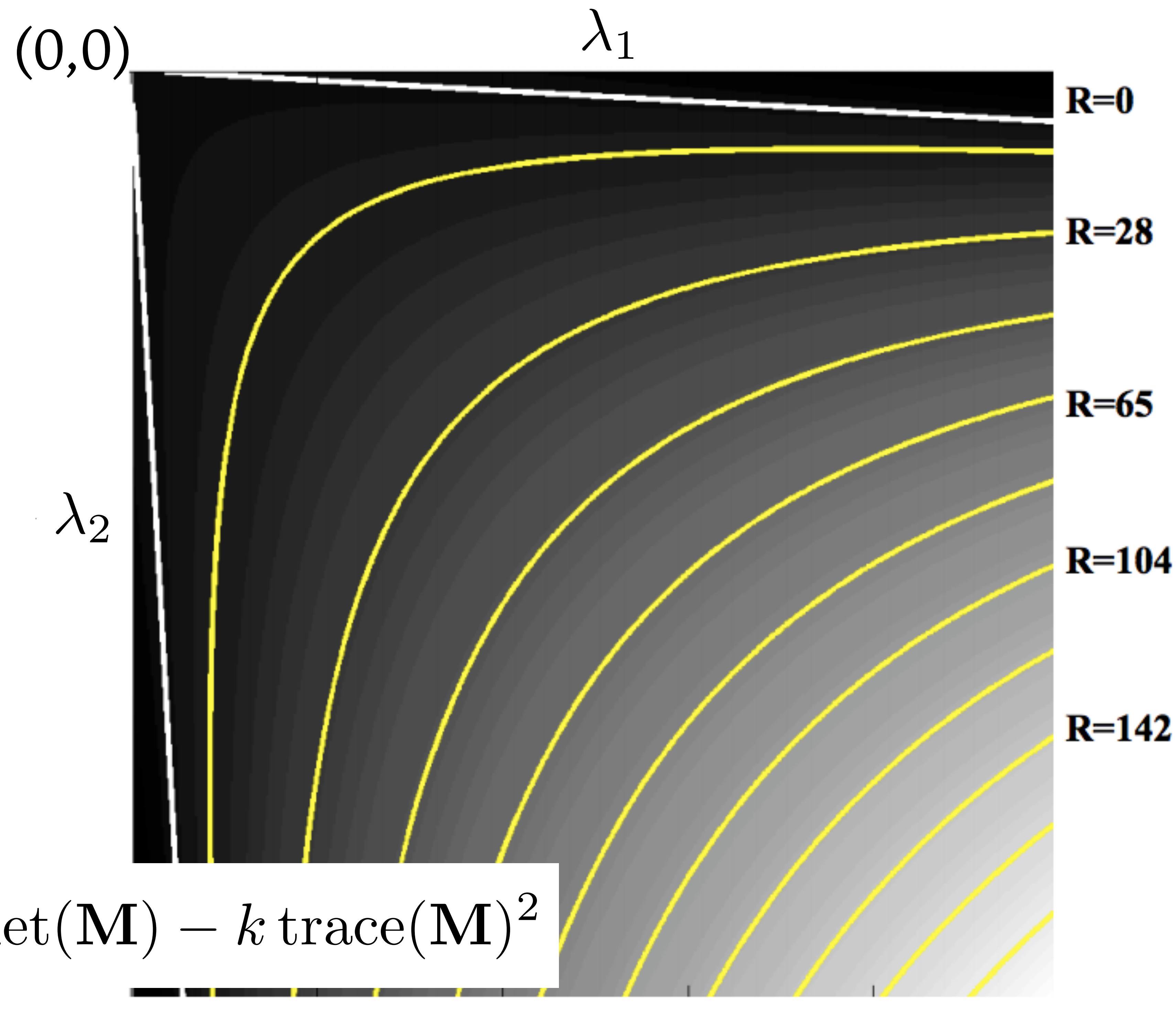
- Corner response ***R***

$$R = \det(\mathbf{M}) - k\,\mathrm{trace}(\mathbf{M})^2$$

$$\det(\mathbf{M}) = \lambda_1 \lambda_2$$

$$\mathrm{trace}(\mathbf{M}) = \lambda_1 + \lambda_2$$

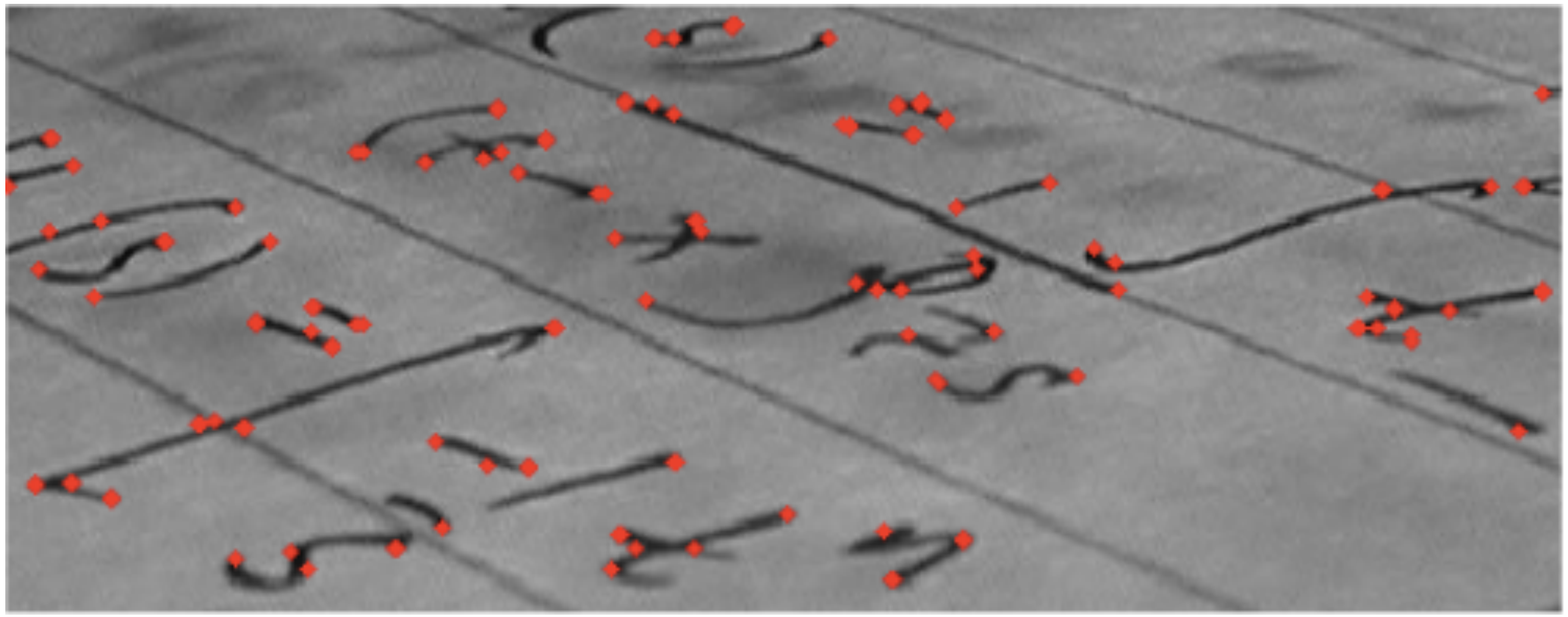
Harris corner detector

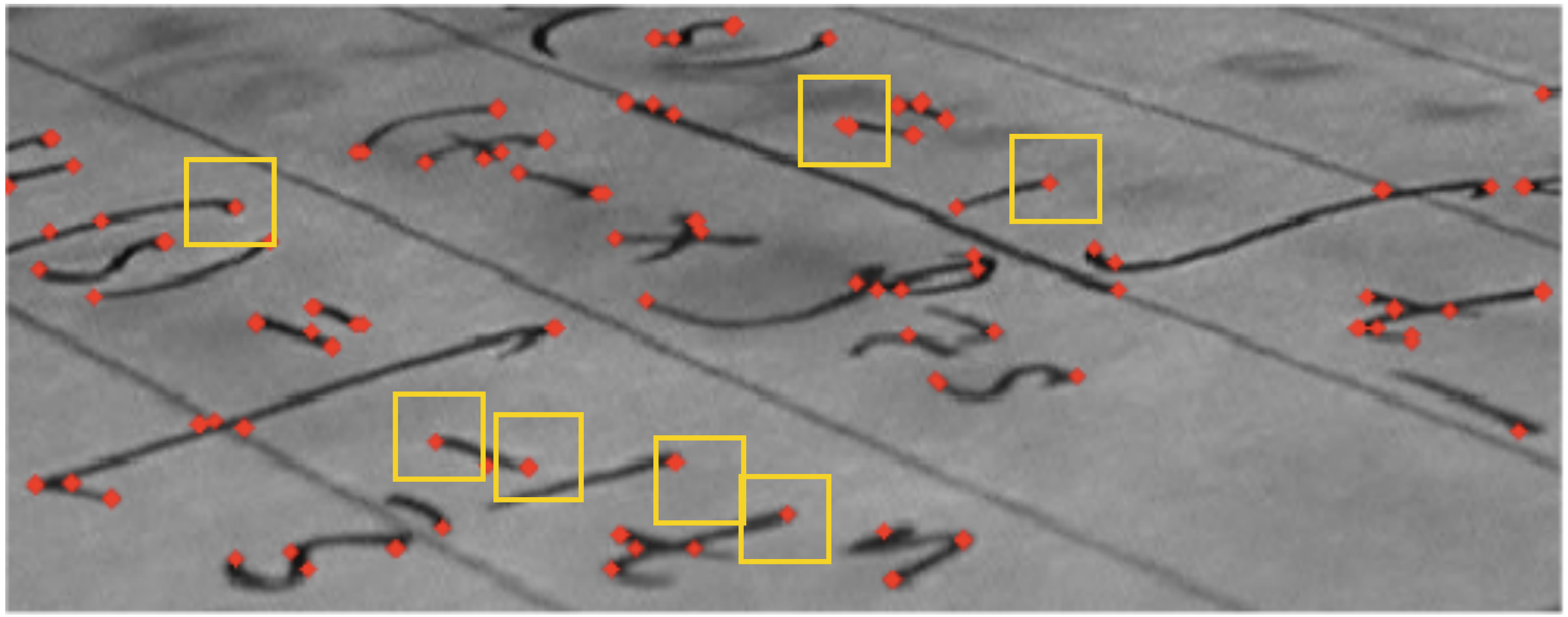


$$R = \det(\mathbf{M}) - k\,\mathrm{trace}(\mathbf{M})^2$$

[source: R. Collins]

Harris corner detector





[source: wikipedia]

Image representation

- There are a large number of image features available for capturing shape, texture or color information
- Most popular ones first detect some '**good points**', and then encode the local information around those points.
- However, all these features are "designed" or hand-crafted. Their performance is limited by their suitability for different tasks.
- In the last two decades, the trend has changed. **Features are now learned rather than designed.**
- There are many different ways to learn features.
    - Autoencoder, for example. You can train an autoencoder with a bunch of image samples, and then use only the encoder part (its output and kernels) for feature extraction.