# Chapter 04
# Interest Point Detection and Matching

Lin ZHANG
School of Computer Science and Technology
Tongji University

# Content

- General requirements for interest point detectors and descriptors
- Harris feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
- SIFT feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
- ORB feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
  - Multiscale processing
- Feature point matching
- Practice

# General requirements for interest point detectors and descriptors

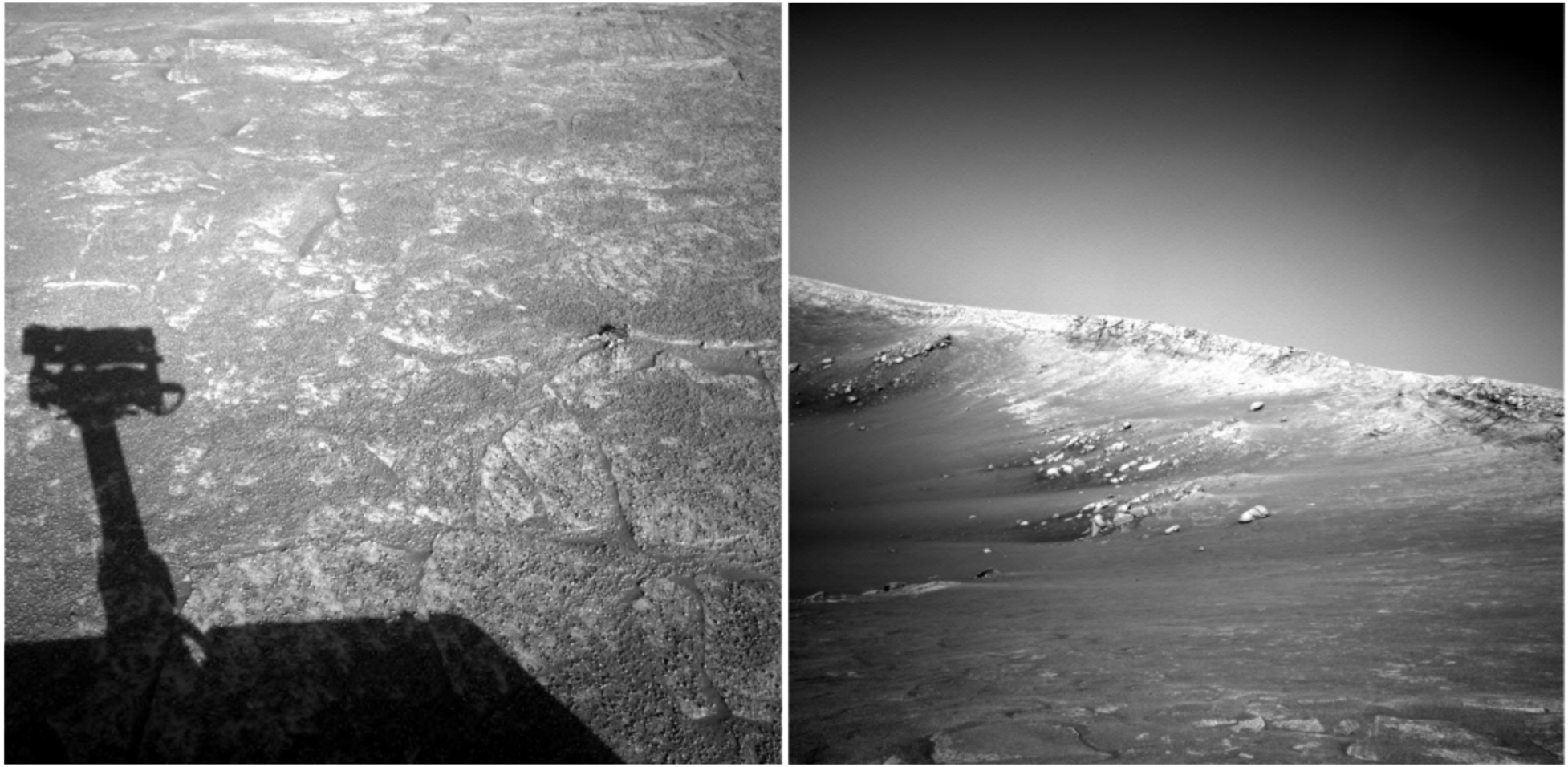
- For point detectors
  - **Locality**: The positions of feature points need to be easily and accurately locatable. For example, edge points in an image do not exhibit good localization because moving along the edge direction passes through points with highly similar appearances
  - **Sparsity**: The number of feature points in an image should be relatively sparse compared to the total number of pixels. If the detected feature points are too dense, it will significantly increase the computational cost of subsequent processing
  - **Stability to illumination changes**: When the environmental lighting conditions change, we expect the feature point detection algorithm to still identify the same feature points
  - **Stability to geometric transformations**: When the camera shooting perspective changes, the image plane undergoes corresponding geometric transformations, and we hope that the feature point detection algorithm can still detect the corresponding feature points

# General requirements for interest point detectors and descriptors

- For point descriptors
  - **High discriminability**: Suppose $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are two image feature points (from different images), and $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ are their respective feature descriptors. If $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ correspond to the same point in the physical scene, we expect $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ to be identical; if $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ correspond to different points in the physical scene, we expect the distance between $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ to be large
  - **Stability to illumination changes**: When the environmental lighting conditions change, we hope that the feature descriptors constructed for the corresponding feature points can remain unchanged
  - **Stability to geometric transformations**: When the camera shooting perspective changes, the image plane undergoes corresponding geometric transformations, and we hope that the feature descriptors constructed for the corresponding feature points can remain unchanged

# General requirements for interest point detectors and descriptors

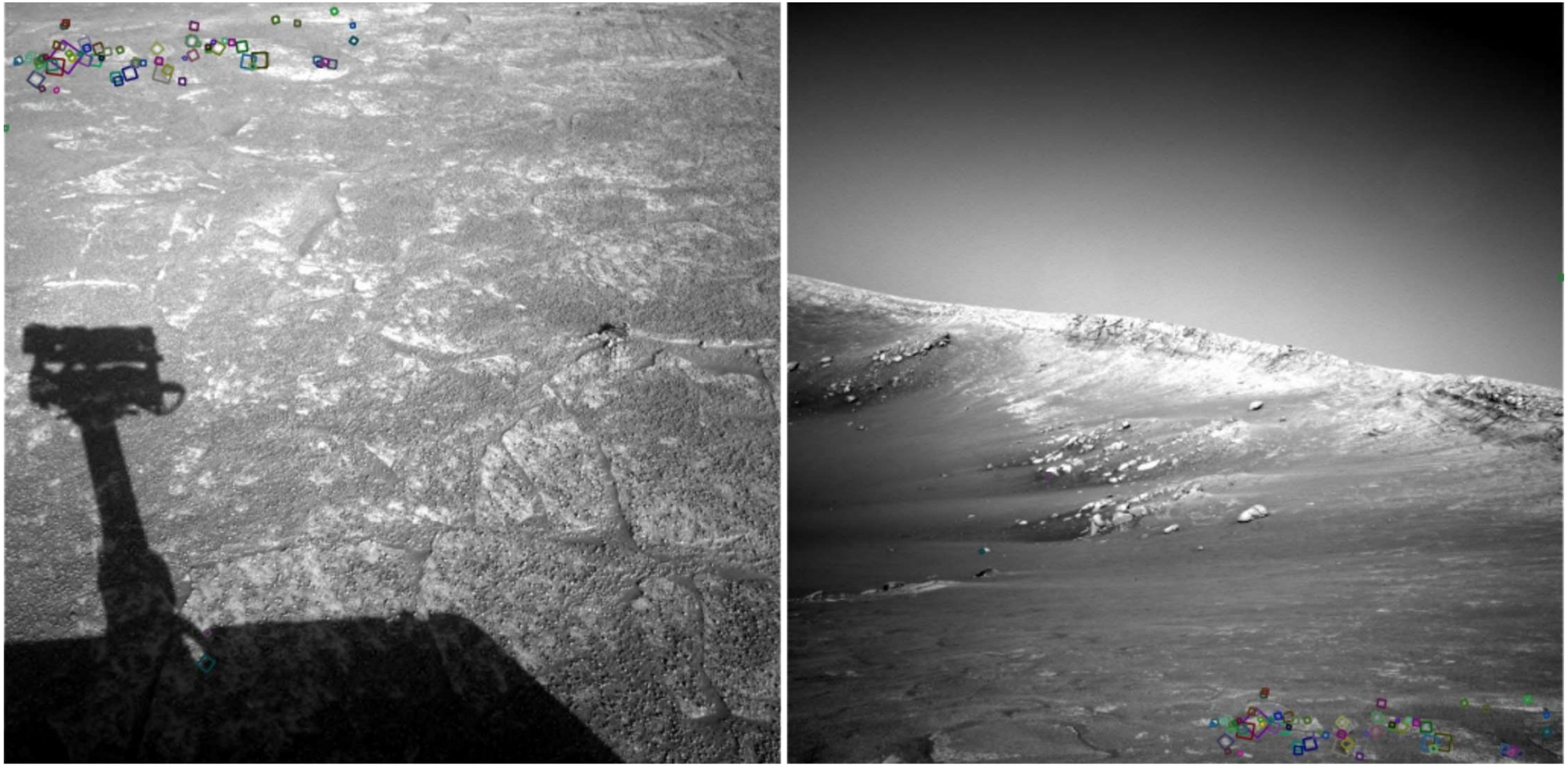
## An example



NASA Mars Rover Images

# General requirements for interest point detectors and descriptors

An example (Look for tiny colored squares)



NASA Mars Rover images with SIFT matches

# Content

- General requirements for interest point detectors and descriptors
- Harris feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
- SIFT feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
- ORB feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
  - Multiscale processing
- Feature point matching
- Practice

# Harris feature—point detector

My office,

5:30PM, Sep. 18, 2011

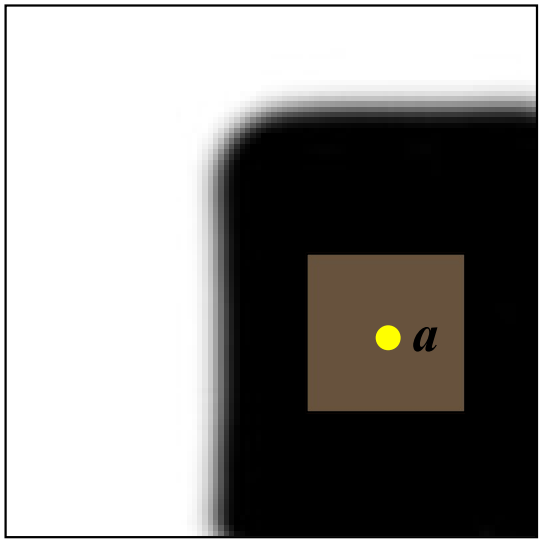
# Harris feature—point detector

- Key property: in the region around a corner, image gradient has two or more dominant directions
- Corners are repeatable and distinctive

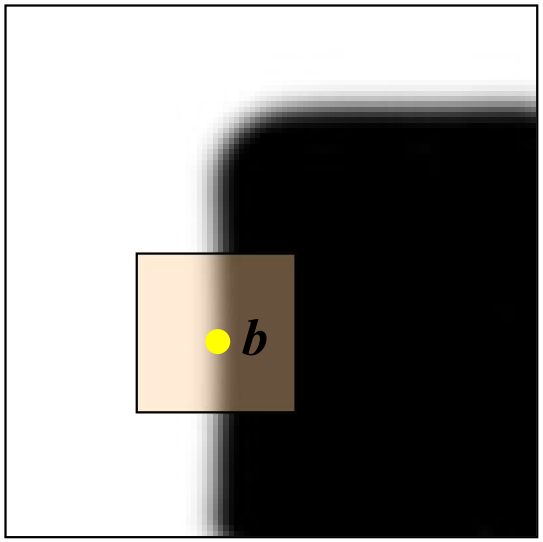
C. Harris and M. Stephens. "A Combined Corner and Edge Detector." *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988

# Harris feature—point detector (basic idea)

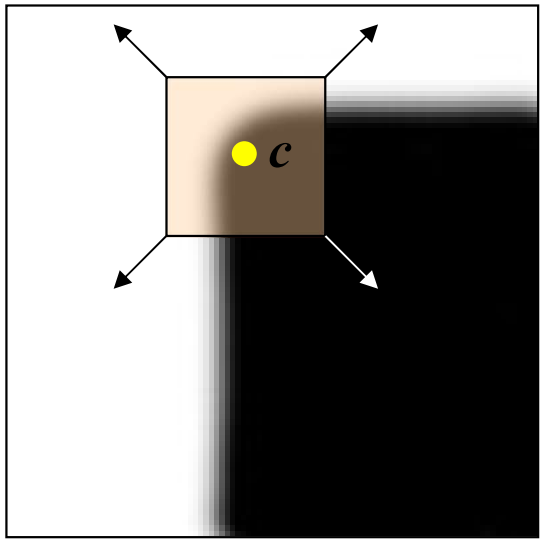
- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity




"flat" region:
no change in all directions

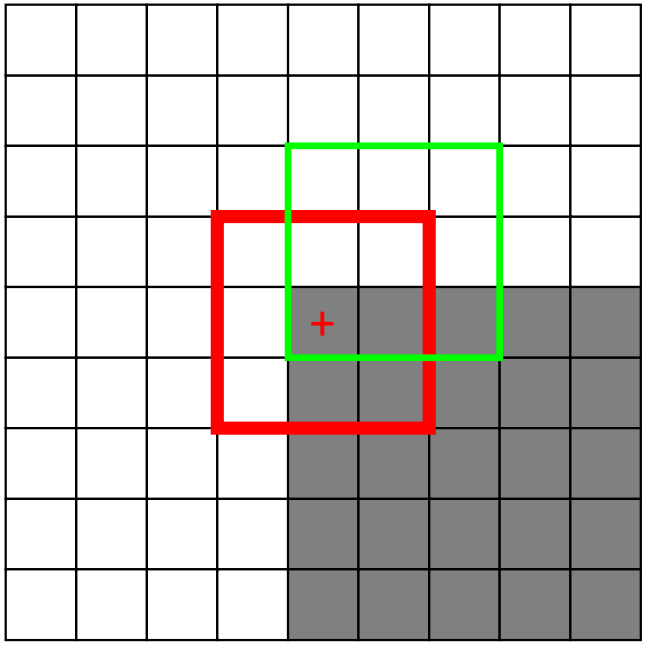


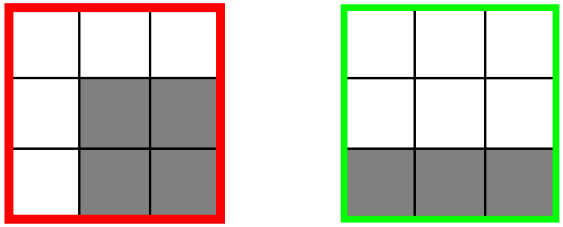

"edge":
no change along the edge direction




"corner":
significant change in all directions

# Harris feature—point detector (basic idea)



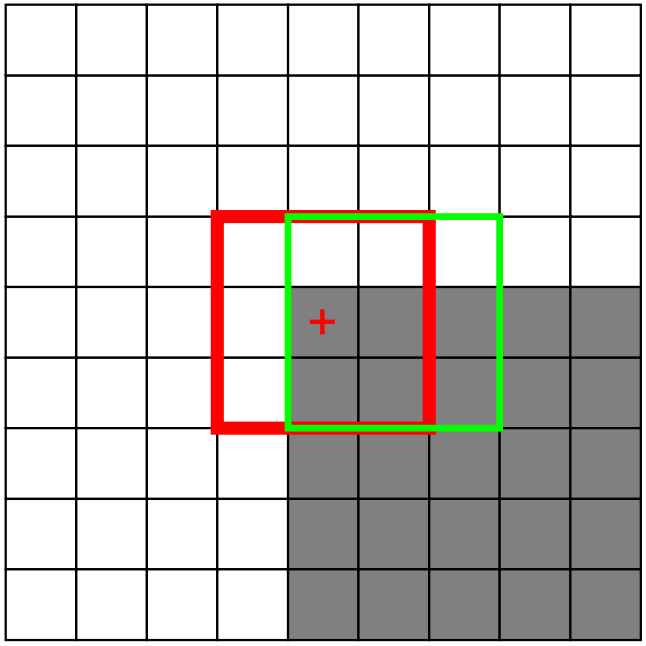



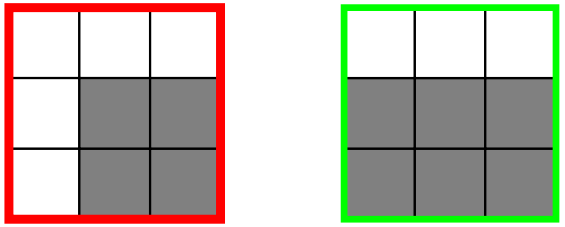
**Difference = 3**

Demo of a point + with well distinguished neighborhood.

**Moving the window in any direction will result in a large intensity change.**

# Harris feature—point detector (basic idea)



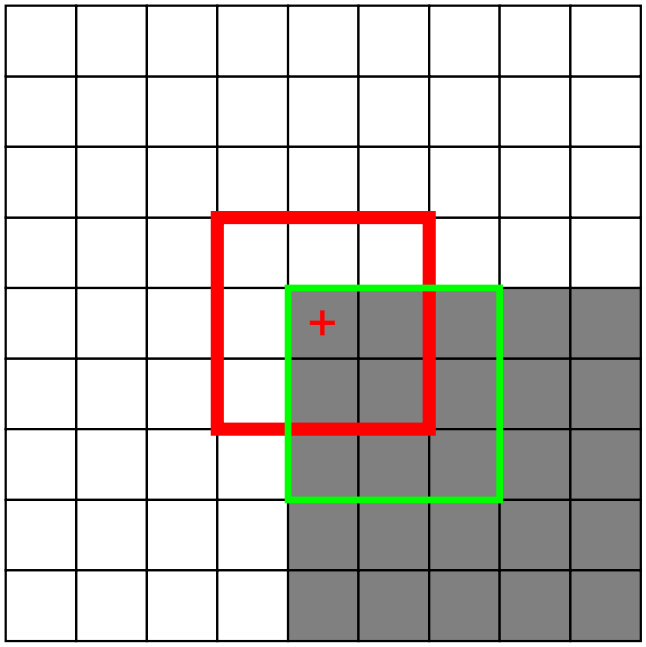



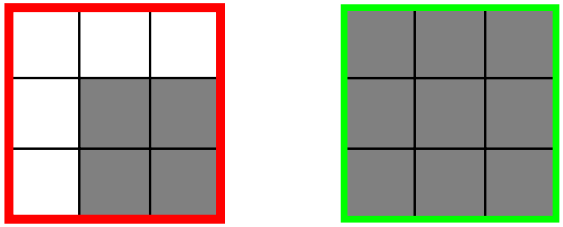
**Difference = 2**

Demo of a point + with well distinguished neighborhood.

**Moving the window in any direction will result in a large intensity change.**

# Harris feature—point detector (basic idea)



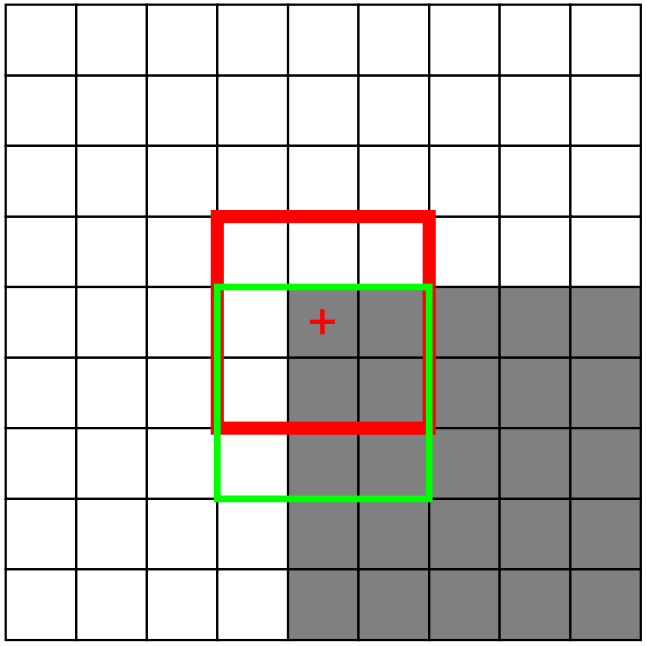



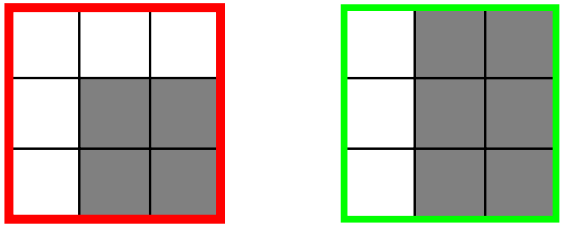
**Difference = 5**

Demo of a point + with well distinguished neighborhood.

**Moving the window in any direction will result in a large intensity change.**

# Harris feature—point detector (basic idea)



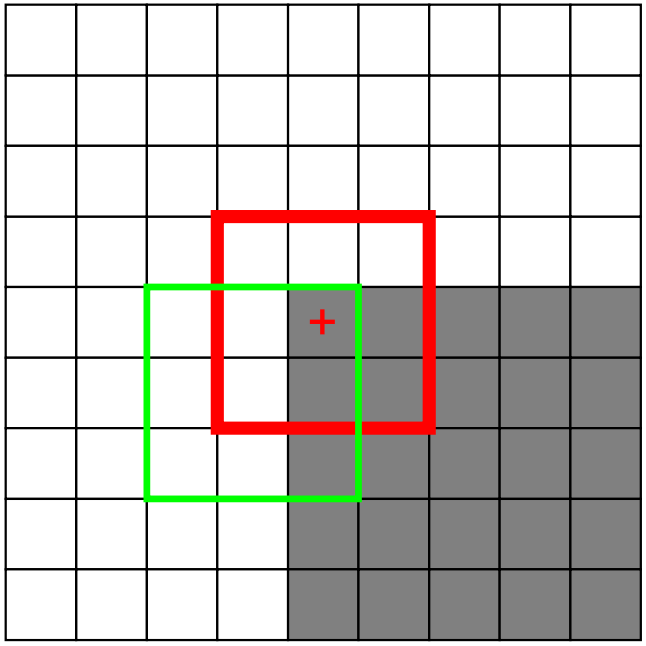


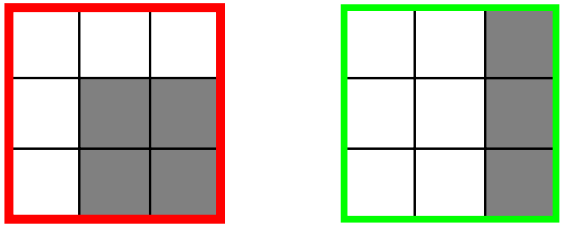
**Difference = 2**

Demo of a point + with well distinguished neighborhood.

**Moving the window in any direction will result in a large intensity change.**

# Harris feature—point detector (basic idea)



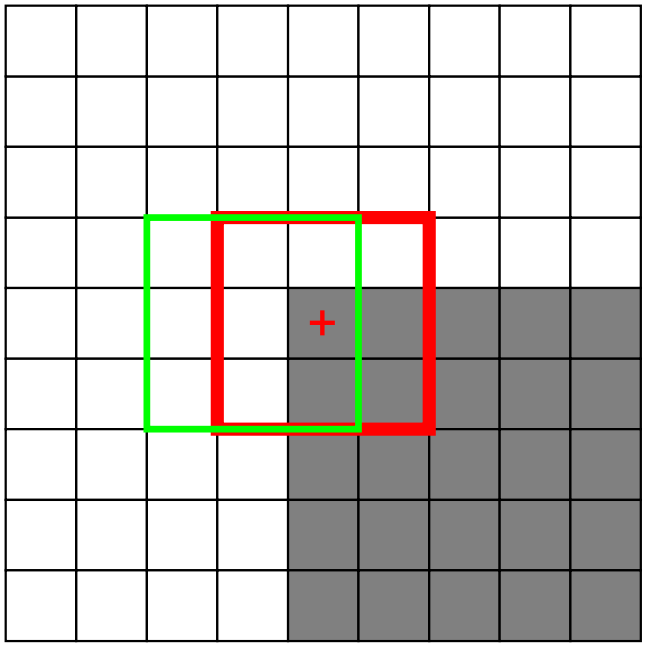


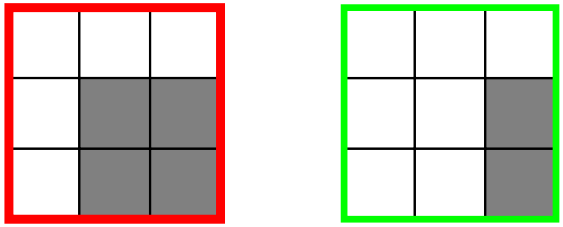
**Difference = 3**

Demo of a point + with well distinguished neighborhood.

**Moving the window in any direction will result in a large intensity change.**

# Harris feature—point detector (basic idea)



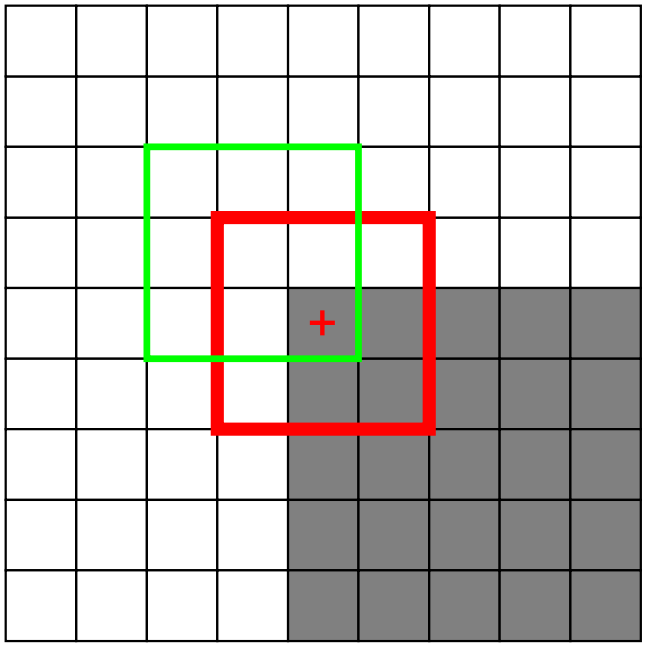


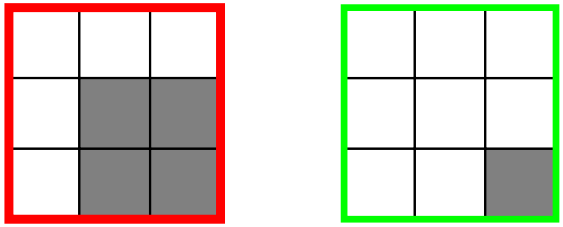
**Difference = 2**

Demo of a point + with well distinguished neighborhood.

**Moving the window in any direction will result in a large intensity change.**

# Harris feature—point detector (basic idea)



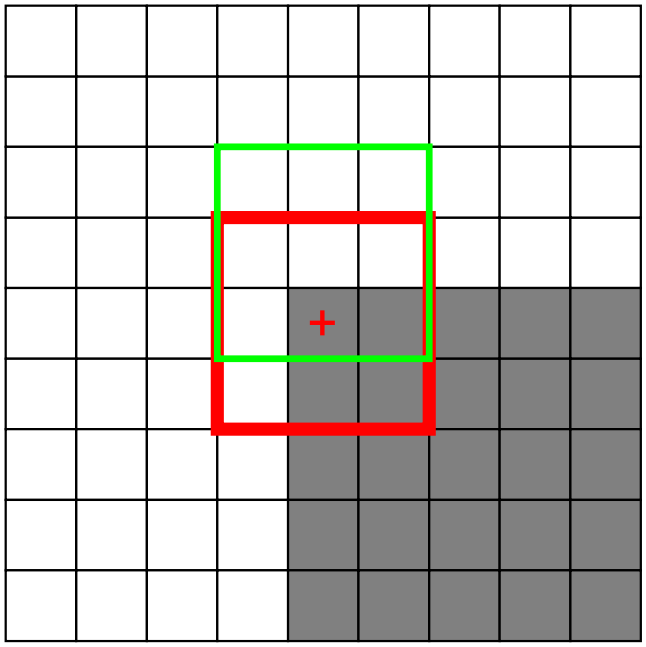



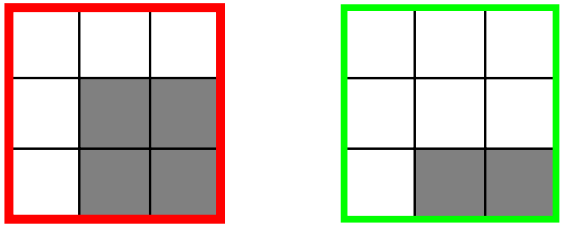
**Difference = 3**

Demo of a point + with well distinguished neighborhood.

**Moving the window in any direction will result in a large intensity change.**

# Harris feature—point detector (basic idea)





**Difference = 2**

Demo of a point + with well distinguished neighborhood.

**Moving the window in any direction will result in a large intensity change.**

# Harris feature—point detector (math)

Consider the image as a 2D function $f(x, y) : \mathbb{R}^2 \rightarrow \mathbb{R}$

Change in appearance of a local patch (defined by a window $W$) centered at $\boldsymbol{x}$ for the shift $(\Delta x, \Delta y)$:

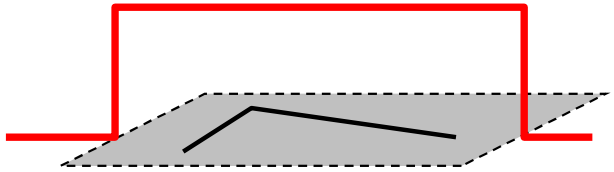
$$s_W(\Delta x, \Delta y) = \sum_{(x_i, y_i) \in W} (f(x_i, y_i) - f(x_i + \Delta x, y_i + \Delta y))^2$$

Window function

Intensity

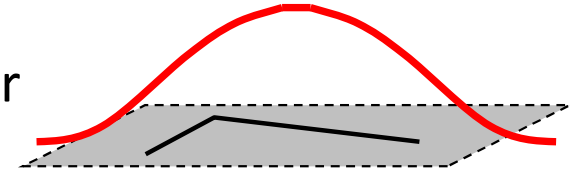
Shifted intensity

Window function $W$=



1 in window, 0 outside

or



Gaussian

# Harris feature—point detector (math)

$$s_W(\Delta x, \Delta y) = \sum_{(x_i, y_i) \in w} (f(x_i, y_i) - f(x_i + \triangle x, y_i + \triangle y))^2 \quad (1)$$

$$\simeq \sum_{(x_i, y_i) \in W} \left( f(x_i, y_i) - f(x_i, y_i) - \left( \frac{\partial f}{\partial x}|_{(x_i, y_i)}, \frac{\partial f}{\partial y}|_{(x_i, y_i)} \right) \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \right)^2 \quad (2)$$

$$= \sum_{(x_i, y_i) \in W} \left( \left( \frac{\partial f}{\partial x}|_{(x_i, y_i)}, \frac{\partial f}{\partial y}|_{(x_i, y_i)} \right) \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \right)^2$$

$$= \sum_{(x_i, y_i) \in W} (\Delta x, \Delta y) \begin{pmatrix} \frac{\partial f}{\partial x}|_{(x_i, y_i)} \\ \frac{\partial f}{\partial y}|_{(x_i, y_i)} \end{pmatrix} \left( \frac{\partial f}{\partial x}|_{(x_i, y_i)}, \frac{\partial f}{\partial y}|_{(x_i, y_i)} \right) \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

$$= (\Delta x, \Delta y) \left\{ \sum_{(x_i, y_i) \in W} \begin{pmatrix} \frac{\partial f}{\partial x}|_{(x_i, y_i)} \\ \frac{\partial f}{\partial y}|_{(x_i, y_i)} \end{pmatrix} \left( \frac{\partial f}{\partial x}|_{(x_i, y_i)}, \frac{\partial f}{\partial y}|_{(x_i, y_i)} \right) \right\} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

$$= (\Delta x, \Delta y) \begin{pmatrix} \sum_{(x_i, y_i) \in W} \left( \frac{\partial f}{\partial x}|_{(x_i, y_i)} \right)^2 & \sum_{(x_i, y_i) \in W} \left( \frac{\partial f}{\partial x}|_{(x_i, y_i)} \right) \left( \frac{\partial f}{\partial y}|_{(x_i, y_i)} \right) \\ \sum_{(x_i, y_i) \in W} \left( \frac{\partial f}{\partial x}|_{(x_i, y_i)} \right) \left( \frac{\partial f}{\partial y}|_{(x_i, y_i)} \right) & \sum_{(x_i, y_i) \in W} \left( \frac{\partial f}{\partial y}|_{(x_i, y_i)} \right)^2 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

$\boldsymbol{M}_{2\times 2}$

# Harris feature—point detector (math)

$$s_W(\triangle x,\triangle y) \cong [\triangle x,\triangle y]\boldsymbol{M}\begin{bmatrix}\triangle x\\ \triangle y\end{bmatrix},$$

where, $$\boldsymbol{M}=\begin{pmatrix}\sum_{(x_i,y_i)\in W}\left(\frac{\partial f}{\partial x}|_{(x_i,y_i)}\right)^2 & \sum_{(x_i,y_i)\in W}\left(\frac{\partial f}{\partial x}|_{(x_i,y_i)}\right)\left(\frac{\partial f}{\partial y}|_{(x_i,y_i)}\right)\\ \sum_{(x_i,y_i)\in W}\left(\frac{\partial f}{\partial x}|_{(x_i,y_i)}\right)\left(\frac{\partial f}{\partial y}|_{(x_i,y_i)}\right) & \sum_{(x_i,y_i)\in W}\left(\frac{\partial f}{\partial y}|_{(x_i,y_i)}\right)^2\end{pmatrix}$$

$s_W(\triangle x,\triangle y)=1$ represents the trace of $(\triangle x,\triangle y)$ , making the appearance change of the local patch $W$ centered at $\boldsymbol{x}$ equal to 1

When the position $\boldsymbol{x}$ and the window $W$ are given, $\boldsymbol{M}$ is determined and the trace determined by the equation $s_W(\triangle x,\triangle y)=1$ is actually totally determined by $\boldsymbol{M}$

# Harris feature—point detector (math)

## Assignment

(1) $\boldsymbol{M}$ can be proved to be a positive semi-definite matrix

(2) In practice, $\boldsymbol{M}$ is positive definite nearly for sure, then $[\Delta x\ \Delta y]\boldsymbol{M}\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix}=1$ represents an ellipse; the length of its semi-major axis is $1/\sqrt{\lambda_2}$ and the length of its semi-minor axis is $1/\sqrt{\lambda_1}$, where $\lambda_1 \geq \lambda_2 > 0$ are the two eigen-values of $\boldsymbol{M}$

**(Hints: for problem (1), read appendix H; for problem (2), read appendix B)**

The shape of such an ellipse is totally determined by $\boldsymbol{M}$

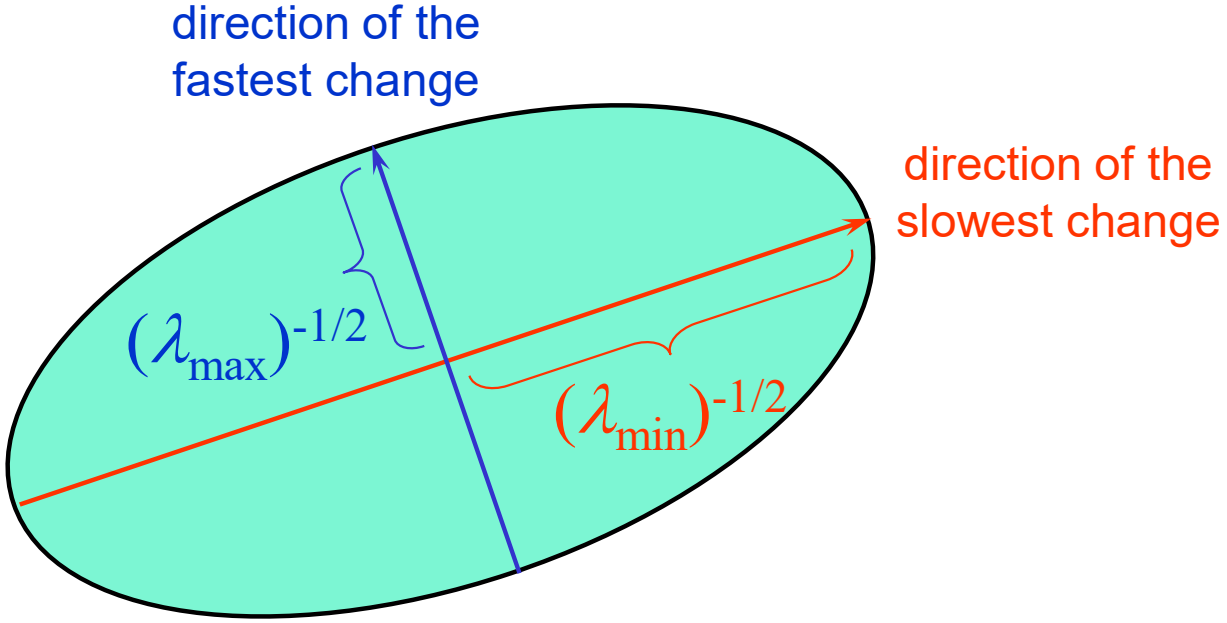
# Harris feature—point detector (math)

Diagonalization of $\boldsymbol{M}$:

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


where $\boldsymbol{R}$ is an orthogonal matrix

The axis lengths of the ellipse are determined by the eigenvalues and the orientation is determined by $\boldsymbol{R}$



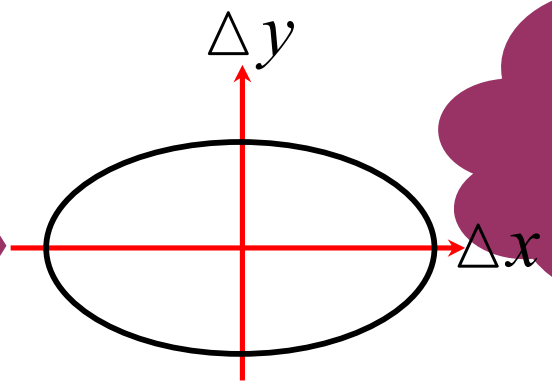
# Harris feature—point detector (math)

The "cornerness" of the window $W$ is reflected by $\boldsymbol{M}$

Suppose there are two local windows $W_1$ and $W_2$; consider the cases when the moving of the two windows leads to the intensity change equals to 1. The moving vector $[\Delta x, \Delta y]$ of each window satisfies the ellipse equation. Thus,

For $W_1$,

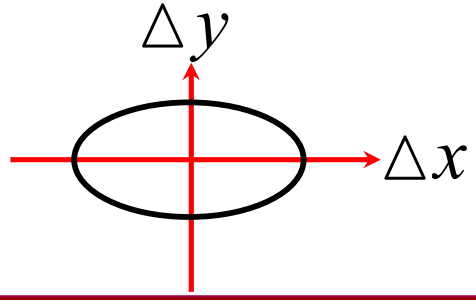
$$[\Delta x, \Delta y]\, \boldsymbol{M}_1 \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = 1$$




*Which window has higher cornerness?*

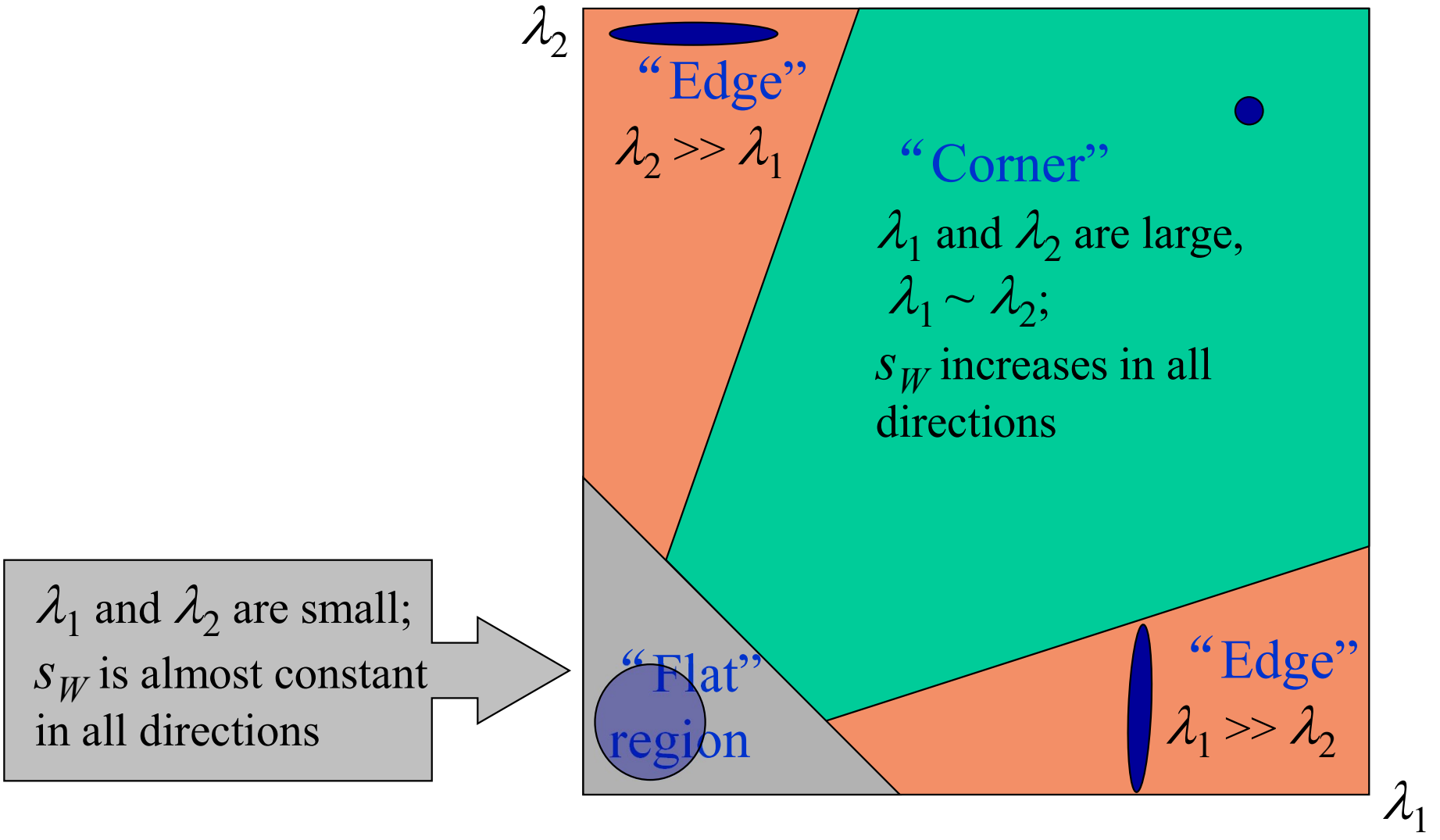
For $W_2$,

$$[\Delta x, \Delta y]\, \boldsymbol{M}_2 \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = 1$$




# Harris feature—point detector (math)

Classification of image points using eigenvalues of $\boldsymbol{M}$:



# Harris feature—point detector (cornerness measure)

Conducting EVD is not implementation efficient, and accordingly, Harris proposed an empirical cornerness measure

Measure of corner response at $\boldsymbol{x}$:

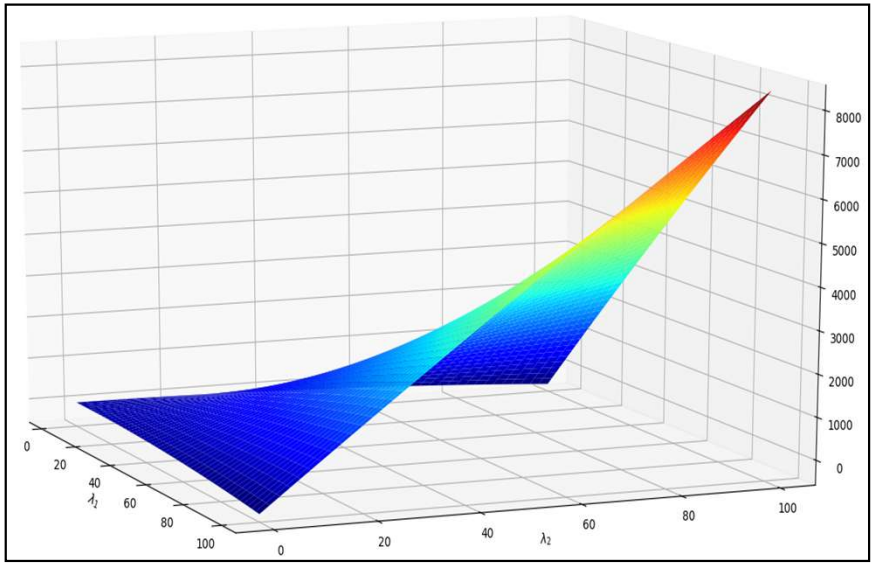
$$r(\boldsymbol{x}) = \det(\boldsymbol{M}(\boldsymbol{x})) - k(trace(\boldsymbol{M}(\boldsymbol{x})))^2$$

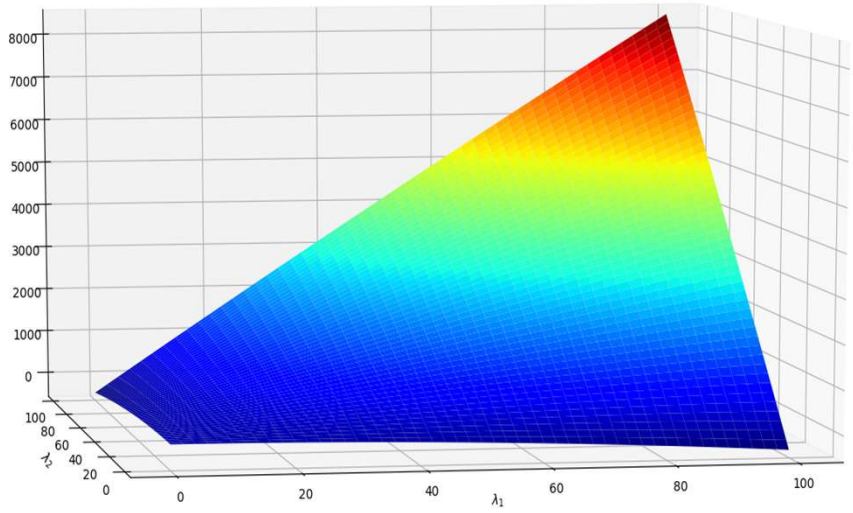
where,

$$\det(\boldsymbol{M}(\boldsymbol{x})) = \lambda_1\lambda_2, trace(\boldsymbol{M}(\boldsymbol{x})) = \lambda_1 + \lambda_2$$

$\lambda_1$ and $\lambda_2$ are the two eigen-values of $\boldsymbol{M}(\boldsymbol{x})$

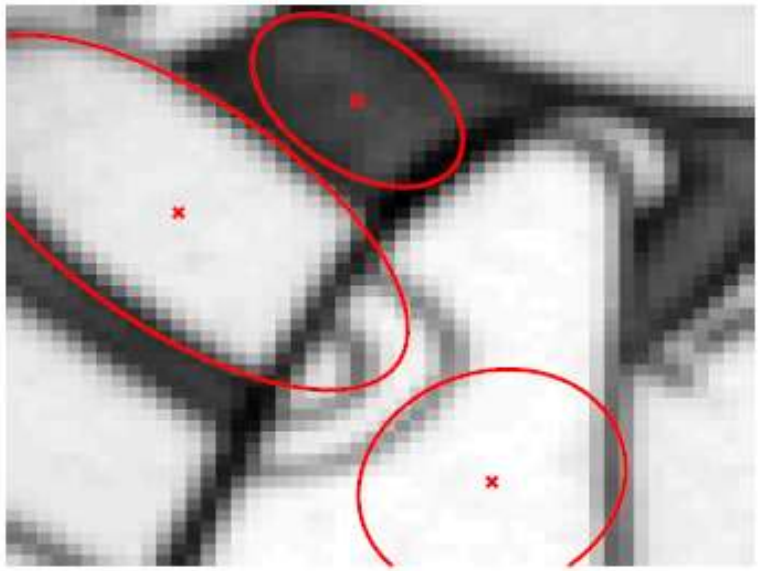
($k$ – empirical constant, $k$= 0.04-0.06)



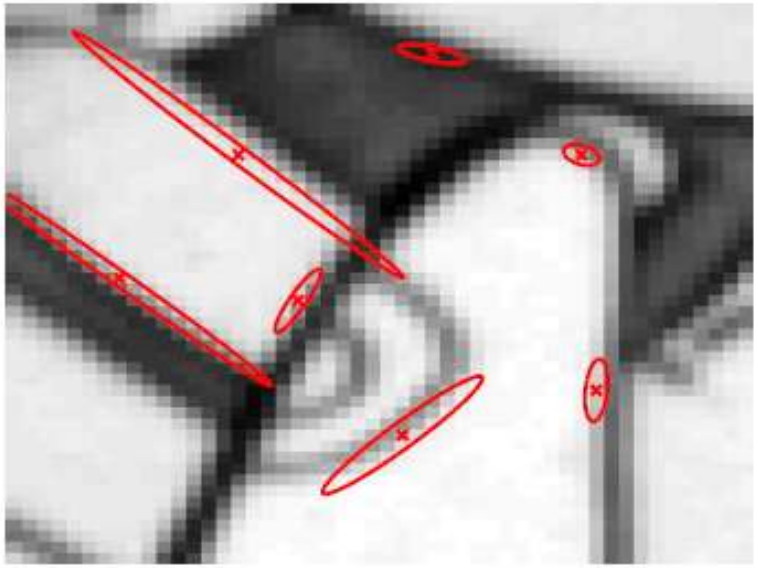



# Harris feature—point detector (illustration)

Ellipse with equation : $[\triangle x, \triangle y]\, \boldsymbol{M} \begin{bmatrix} \triangle x \\ \triangle y \end{bmatrix} = 1$



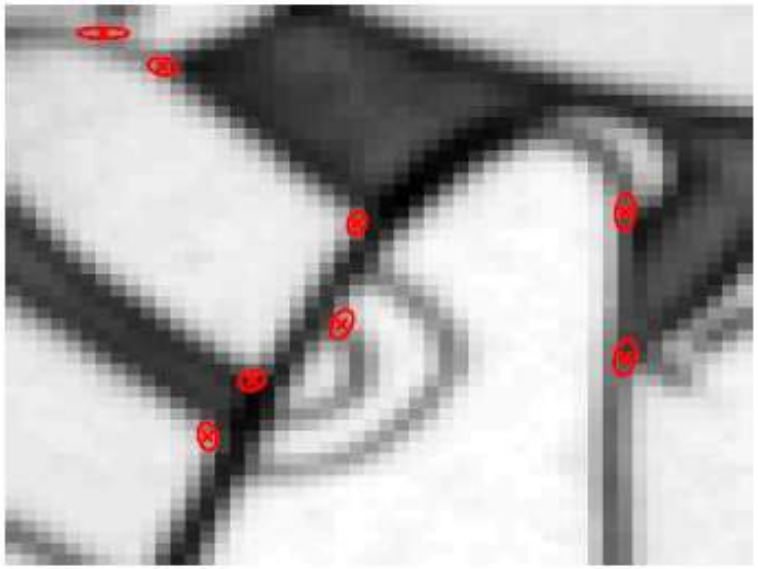
flat region
both eigenvalues small



edge
one small, one large

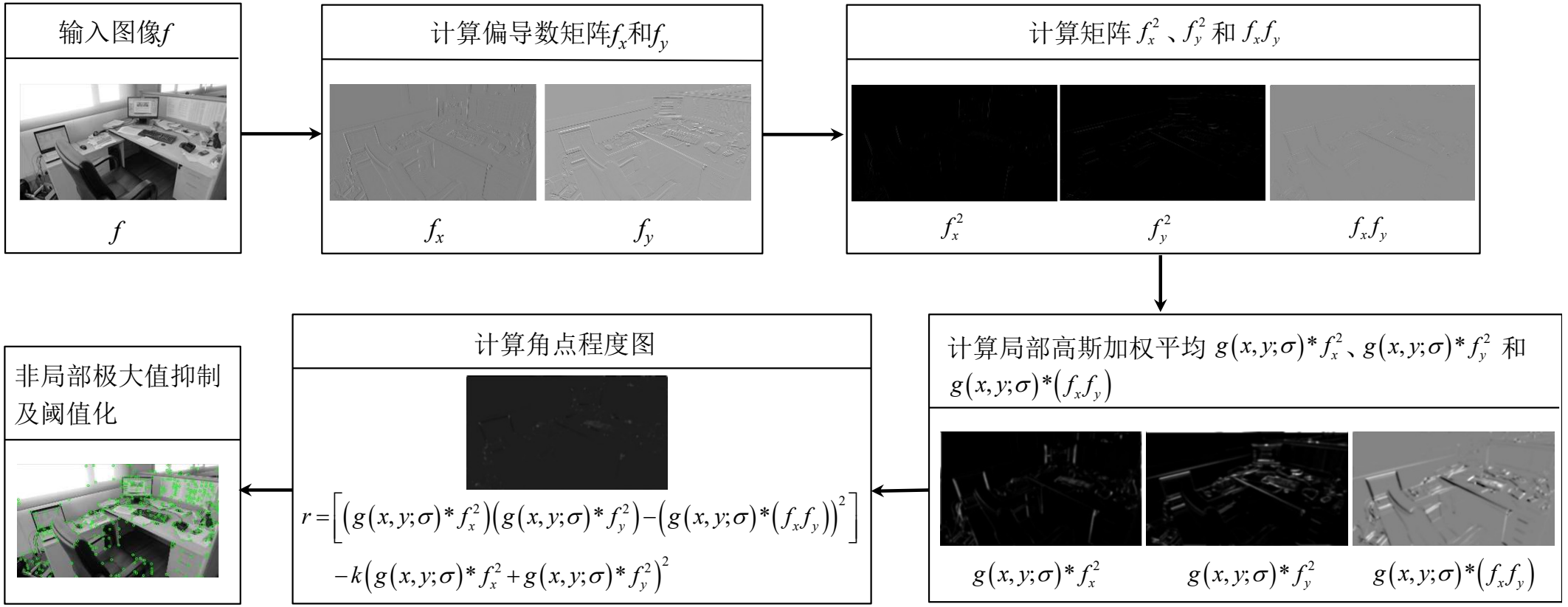
# Harris feature—point detector (illustration)

Ellipse with equation : $[\Delta x, \Delta y]\boldsymbol{M}\begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix} = 1$



corner
both eigenvalues large

# Harris feature—point detector (implementation)

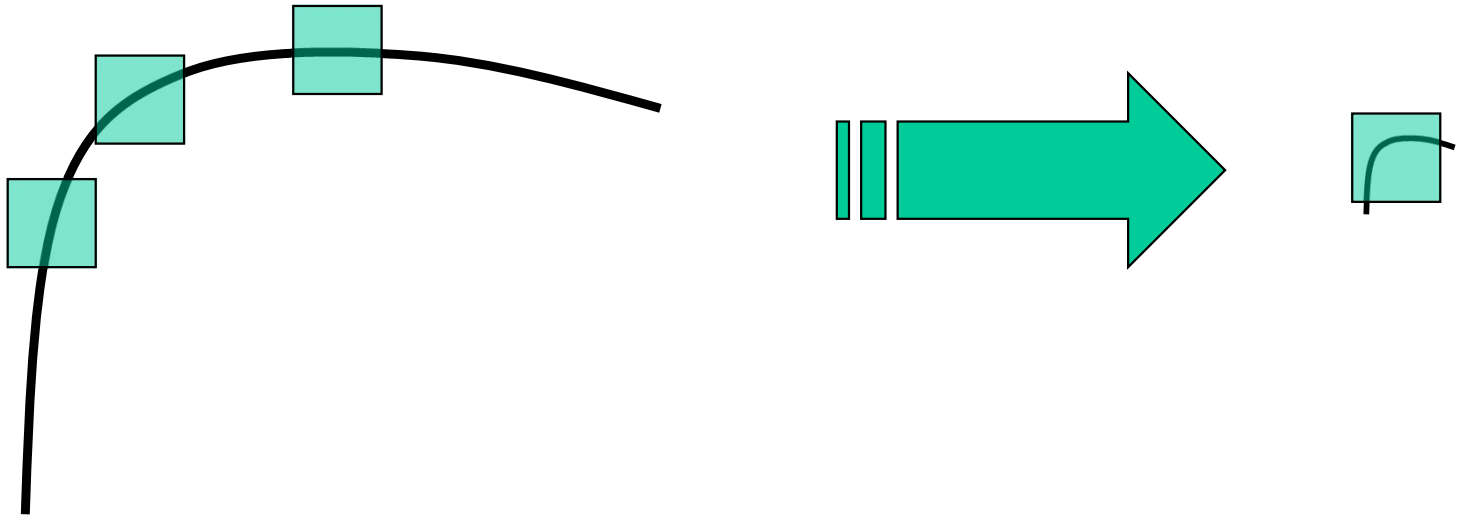



where $g(x,y;\sigma)=\frac{1}{2\pi\sigma^2}\exp\left(-\frac{\left(x^2+y^2\right)}{2\sigma^2}\right)$ is a 2D Gaussian window, with the side-length $6\sigma$

Note: For how to compute $f_x$ and $f_y$, please refer to Appendix C of the textbook

# Harris feature—point detector (invariance)

- From the definition of the corner, theoretically, Harris corner detector is rotation invariant
- **However, it is not invariant to scale change!**



All points will be classified as edges

Corner !

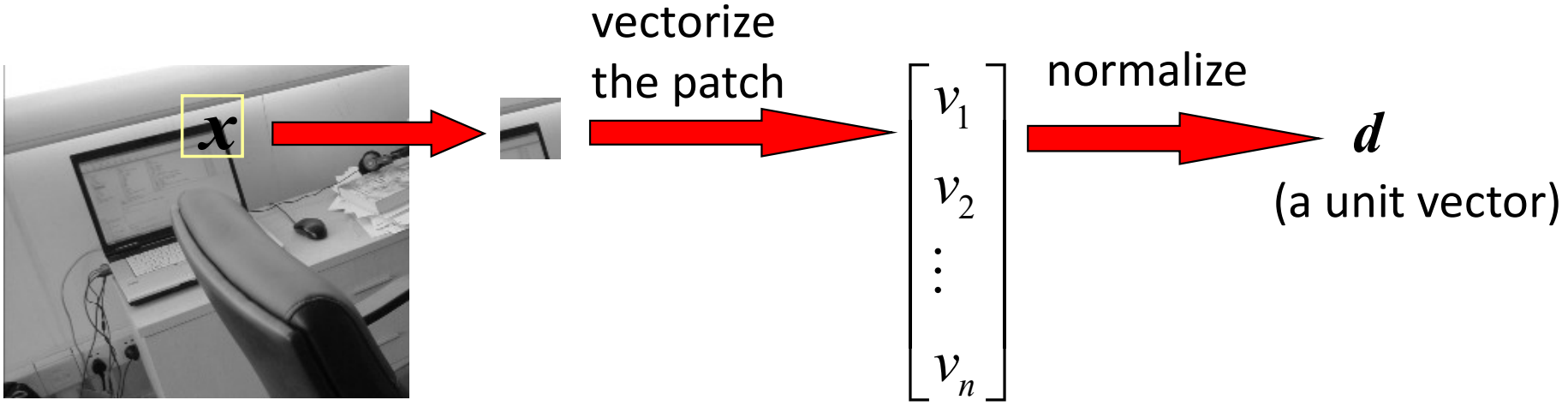
The underlying reason is that Harris corner detection scheme does not provide an automatic and appropriate window size selection method!

# Content

- General requirements for interest point detectors and descriptors
- Harris feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
- SIFT feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
- ORB feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
  - Multiscale processing
- Feature point matching
- Practice

# Harris feature—feature descriptor

- "Block" descriptor for a Harris corner $\boldsymbol{x}$
  - Take a region with a fixed size around $\boldsymbol{x}$
  - Stack the region into a vector and normalize it as $\boldsymbol{d}$
  - This vector $\boldsymbol{d}$ serves as the descriptor for $\boldsymbol{x}$



# Harris feature—feature descriptor

- "Block" descriptor for a Harris corner $\boldsymbol{x}$
  - Take a region with a fixed size around $\boldsymbol{x}$
  - Stack the region into a vector and normalize it as $\boldsymbol{d}$
  - This vector $\boldsymbol{d}$ serves as the descriptor for $\boldsymbol{x}$
- Deficiencies of such simple descriptors
  - Not rotation invariant
  - Not scale invariant

# Harris feature—feature descriptor

- "Block" descriptor for a Harris corner $\boldsymbol{x}$
  - Take a region with a fixed size around $\boldsymbol{x}$
  - Stack the region into a vector and normalize it as $\boldsymbol{d}$
  - This vector $\boldsymbol{d}$ serves as the descriptor for $\boldsymbol{x}$
- We want:
  - **Rotation and scale invariant feature points**
  - **Rotation and scale invariant feature descriptors**

That motivates us to seek more advanced feature point detectors and feature point descriptors

# Content

- General requirements for interest point detectors and descriptors
- Harris feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
- SIFT feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
- ORB feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
  - Multiscale processing
- Feature point matching
- Practice

# Harris feature—distance between two descriptors

Given two (block) descriptors $\boldsymbol{d}_1, \boldsymbol{d}_2 \in \mathbb{R}^n$, their distance can be computed in different ways

(1) Sum of squared differences (SSD): $SSD_{dist}\left(\boldsymbol{d}_1, \boldsymbol{d}_2\right) = \left\|\boldsymbol{d}_1 - \boldsymbol{d}_2\right\|_2^2 = \sum_{i=1}^{n}\left(d_1^i - d_2^i\right)^2$

(2) Sum of absolute differences (SAD): $SAD_{dist}\left(\boldsymbol{d}_1, \boldsymbol{d}_2\right) = \sum_{i=1}^{n}\left|d_1^i - d_2^i\right|$

(3) Normalized cross correlation (NCC):

$$NCC_{dist}\left(\boldsymbol{d}_1, \boldsymbol{d}_2\right) = 1 - \frac{1}{n}\frac{\left(\boldsymbol{d}_1 - \mu\left(\boldsymbol{d}_1\right)\right)\cdot\left(\boldsymbol{d}_2 - \mu\left(\boldsymbol{d}_2\right)\right)}{std\left(\boldsymbol{d}_1\right)std\left(\boldsymbol{d}_2\right)}$$

or

$$NCC_{dist}\left(\boldsymbol{d}_1, \boldsymbol{d}_2\right) = \arccos\left(\frac{1}{n}\frac{\left(\boldsymbol{d}_1 - \mu\left(\boldsymbol{d}_1\right)\right)\cdot\left(\boldsymbol{d}_2 - \mu\left(\boldsymbol{d}_2\right)\right)}{std\left(\boldsymbol{d}_1\right)std\left(\boldsymbol{d}_2\right)}\right)$$

where $\mu\left(\boldsymbol{d}_1\right)$ returns the mean value of $\boldsymbol{d}_1$ and $std(\boldsymbol{d}_1)$ returns the standard deviation of $\boldsymbol{d}_1$

# Content

- General requirements for interest point detectors and descriptors
- Harris feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
- SIFT feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
- ORB feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
  - Multiscale processing
- Feature point matching
- Practice

# SIFT feature—overview

- Deficiencies of the Harris feature
  - Harris corner detector is not scale invariant since it does not have an automatic size determination strategy for the local window $W$
  - "block" descriptor is not scale invariant since the size of its local window is set empirically

⬇

We expect **scale invariant feature points** and **scale invariant feature descriptors**

⬇

- SIFT feature can meet our needs
  - ✓ It has a scale invariant feature point detection scheme
  - ✓ It has a scale invariant feature descriptor construction scheme

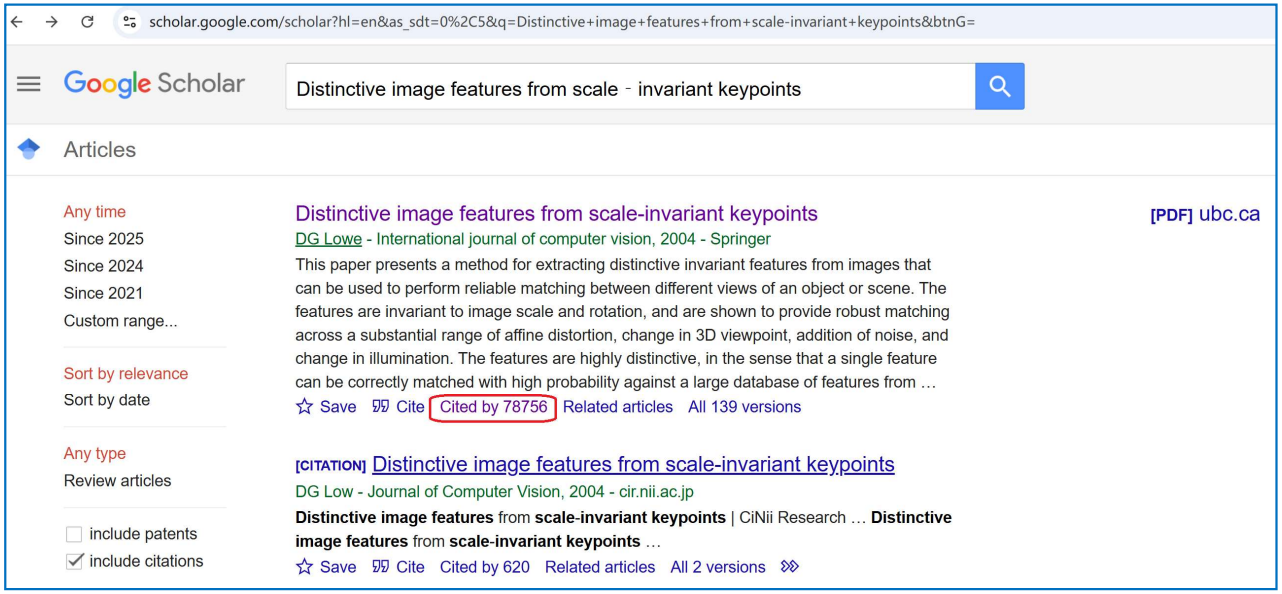
# SIFT feature—overview

- Point detector in SIFT
  - It uses a series of scale-normalized LoGs with different scales to detect blobs in the given image in a scale-invariant manner
  - When a blob is detected, a byproduct is also obtained, its **characteristic scale (scale covariant)**, which will be used to build the feature descriptor for the blob center
- Feature descriptor in SIFT
  - For a blob center, its feature descriptor is constructed based on its characteristic scale and thus it is scale-invariant

# SIFT feature—overview

Prof. David Lowe. From 2015-2018, David Lowe was a Senior Research Scientist with Google in the Machine Intelligence Group. From 2009-2015 he was co-founder and chairman of Cloudburst Research, a computer vision startup in Vancouver that was acquired by Google in May 2015. From 1987-2015, he was a professor of computer science at the University of British Columbia

D.G. Lowe, Distinctive image features from scale-invariant keypoints, *IJCV* 60 (2), pp. 91-110, 2004




This paper has been cited by 78756 times by the day May 19, 2025

# SIFT feature—point detector (basic idea)

- In SIFT, the feature point is the “blob center”
  - To characterize a blob, we need to know its center position and also its spatial extension (or its size)

# SIFT feature—point detector (basic idea)

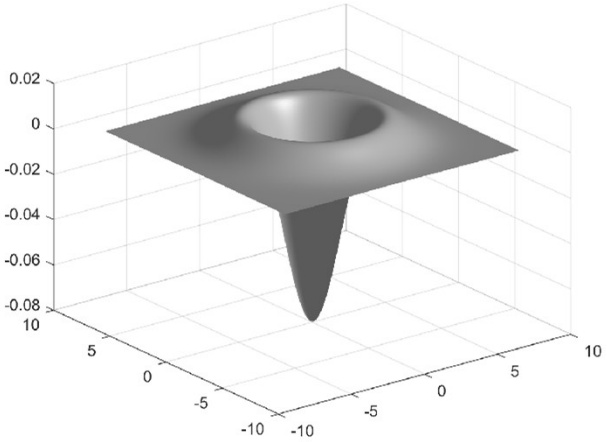
Tony Linderberg proposes to use **scale-normalized LoG** to detect blobs[1]

2D isotropic Gaussian function, $g(x,y)=\frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$

LoG (Laplacian of Gaussian), $\nabla^2 g(x,y)=\frac{\partial^2 g}{\partial x^2}+\frac{\partial^2 g}{\partial y^2}=\frac{x^2+y^2-2\sigma^2}{2\pi\sigma^6}e^{-\frac{x^2+y^2}{2\sigma^2}}$

**Scale-normalized LoG**,

$$\sigma^2\nabla^2 g(x,y)=\frac{x^2+y^2-2\sigma^2}{2\pi\sigma^4}e^{-\frac{x^2+y^2}{2\sigma^2}}$$

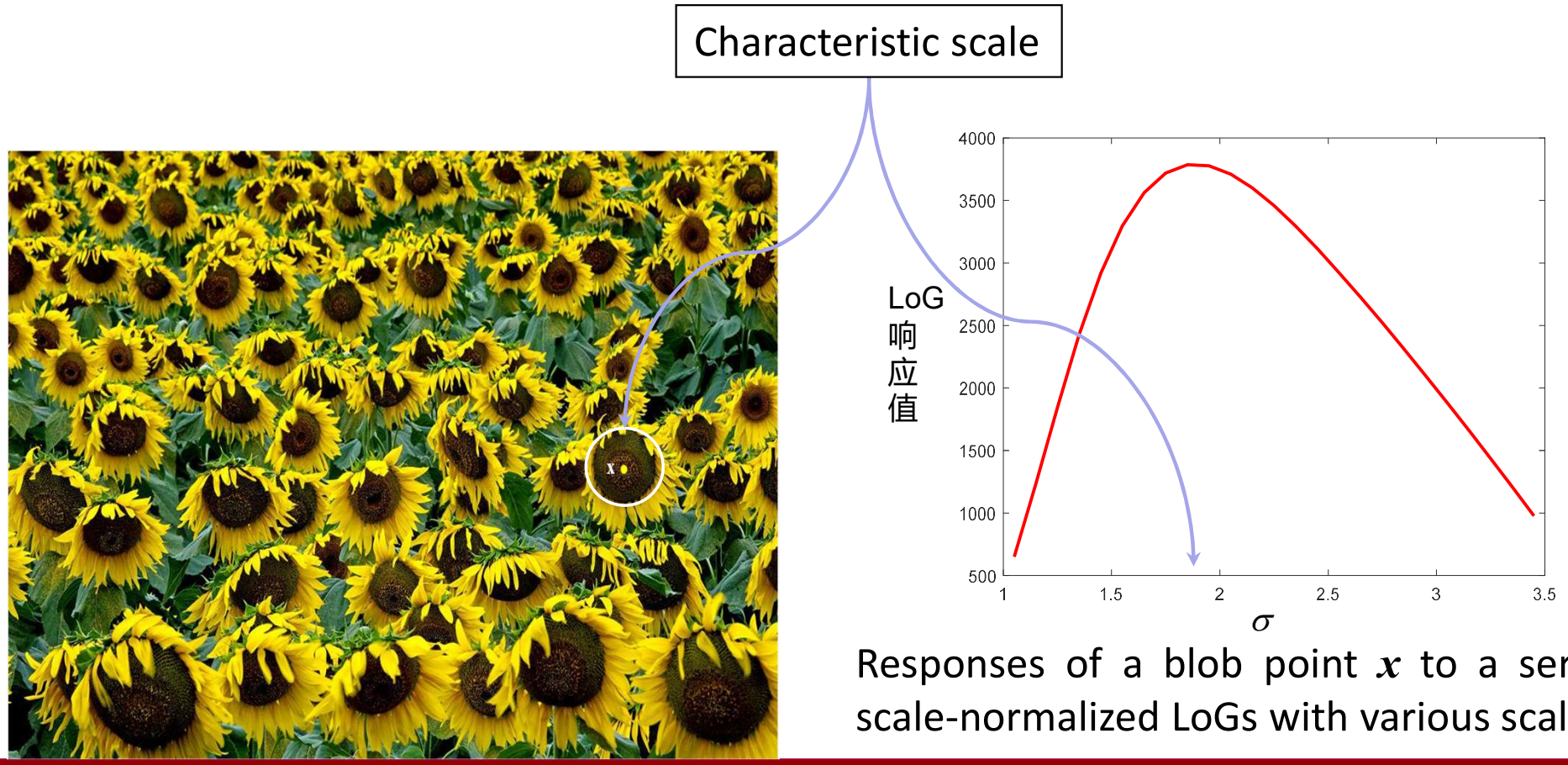



➡ It is quite suitable for **blob** structure detection

[1] T. Linderberg, Scale-space theory: A basic tool for analysing structures at different scales, J. Applied Statistics, 1994, 21(2): 224-270

# SIFT feature—point detector (basic idea)

- For blob detection, we use a “template matching” scheme
- For a real blob point $x$, its responses to scale-normalized LoGs with various scales has a **unique peak** (valley) , and thus we can use the “peak scale” to determine its size, that is, its **characteristic scale**




Responses of a blob point $x$ to a series of scale-normalized LoGs with various scales

# SIFT feature—point detector (basic idea)

The responses of image $\boldsymbol{I}$ at $\boldsymbol{x}$ to the scale-normalized LoGs $\left\{\sigma_i^2\nabla^2 g(\sigma_i)\right\}_{i=1}^{n}$ are $\left\{\left(\sigma_i^2\nabla^2 g(\sigma_i) * \boldsymbol{I}\right)(\boldsymbol{x})\right\}_{i=1}^{n}$

Let $\sigma_{\boldsymbol{x}}^*$ be,

$$\sigma_{\boldsymbol{x}}^* = \arg\max_{\sigma_i}\left(\sigma_i^2\nabla^2 g(\sigma_i) * \boldsymbol{I}\right)(\boldsymbol{x}), i = 1,\ldots,n$$

If $r(\boldsymbol{x}) \triangleq \left(\sigma_{\boldsymbol{x}}^{*2}\nabla^2 g(\sigma_{\boldsymbol{x}}^*) * \boldsymbol{I}\right)(\boldsymbol{x})$ is a unique peak and $\forall \boldsymbol{y} \in \boldsymbol{x}$'s neighborhood,

$$r(\boldsymbol{x}) > r(\boldsymbol{y})$$

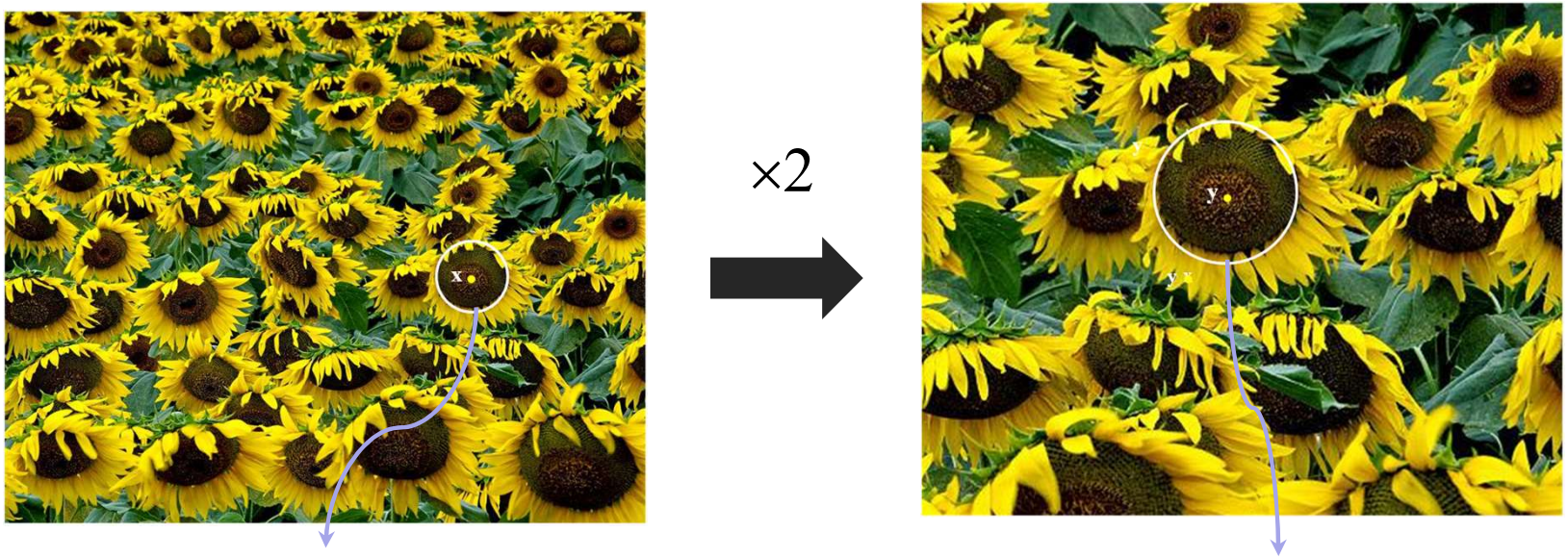
Then we say $\boldsymbol{x}$ is a feature point (blob center) and $\sigma_{\boldsymbol{x}}^*$ is its characteristic scale

Note: for the above statements, changing "peak", "argmax", and ">" to "valley", "argmin", and "<", the conclusion also holds

# SIFT feature—point detector (characteristic scale)

For a blob, its characteristic scale defined above is **scale-covariant**

An example



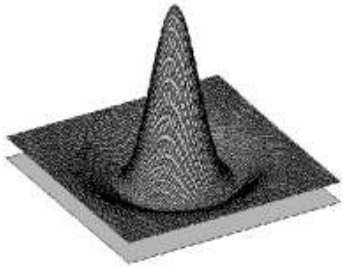

***x***'s characteristic scale obtained by scale-normalized LoGs is 1.5

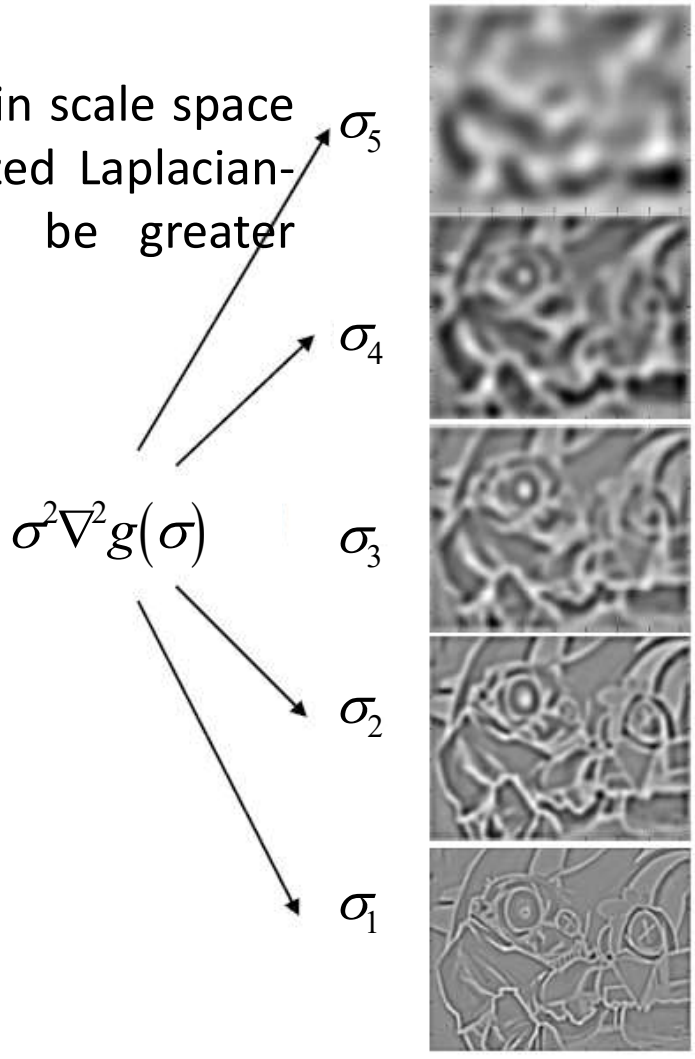
***y***'s characteristic scale obtained by scale-normalized LoGs is 3.0

scale-normalized LoG satisfies our requirement of automatic scale selection for building **scale-invariant feature descriptors**

# SIFT feature—point detector (implementation)

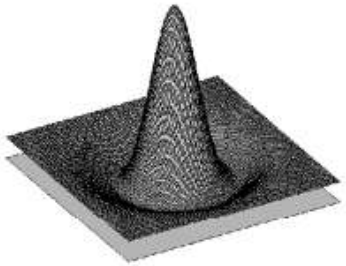
- Interest points:
  - Local extremum in scale space of scale-normalized Laplacian-of-Gaussian and be greater than a threshold

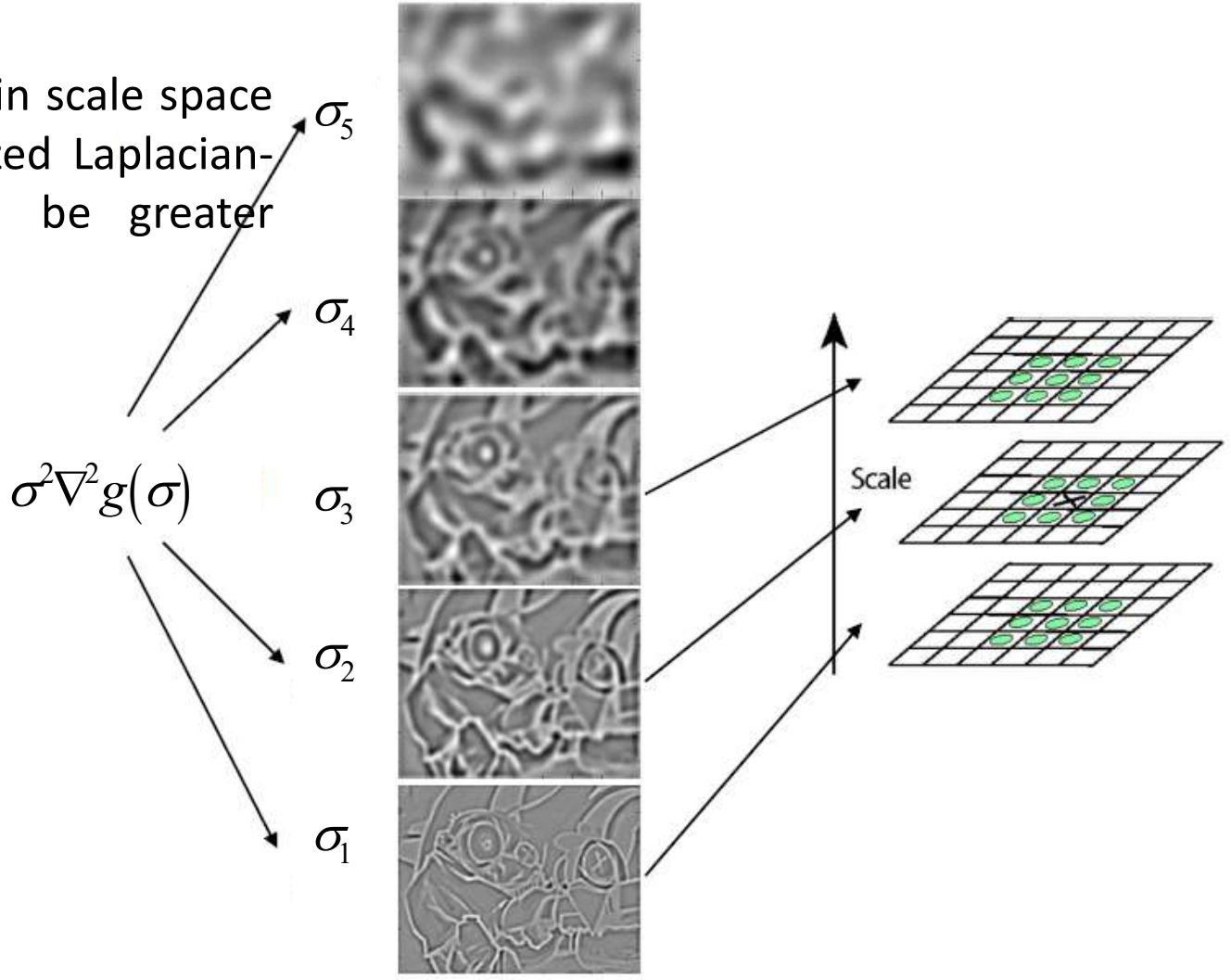




# SIFT feature—point detector (implementation)

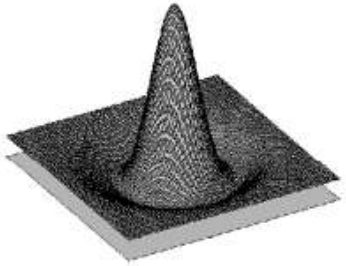
- Interest points:
  - Local extremum in scale space of scale-normalized Laplacian-of-Gaussian and be greater than a threshold

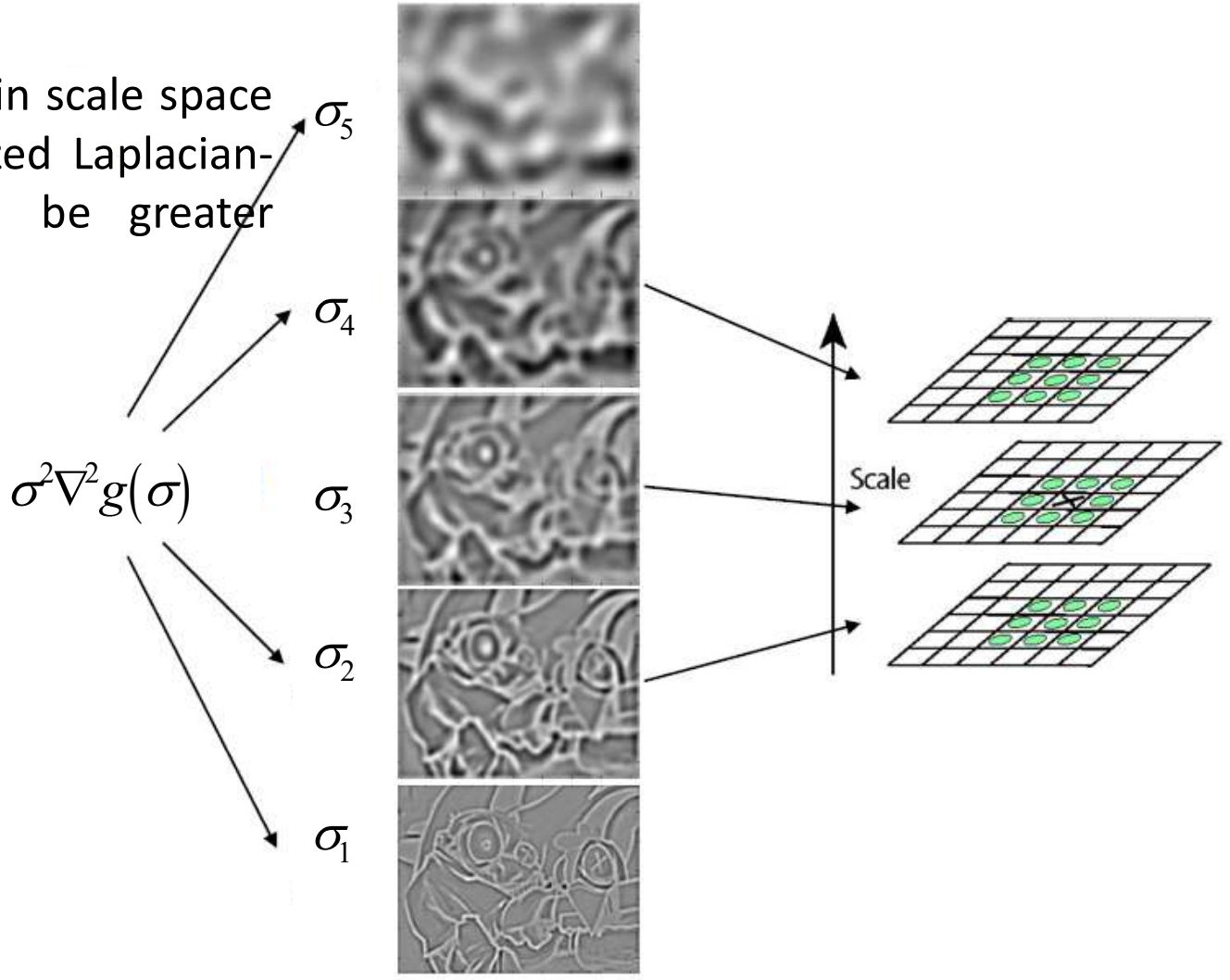




# SIFT feature—point detector (implementation)

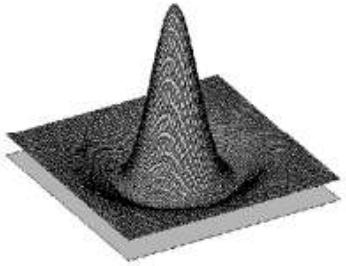
- Interest points:
  - Local extremum in scale space of scale-normalized Laplacian-of-Gaussian and be greater than a threshold

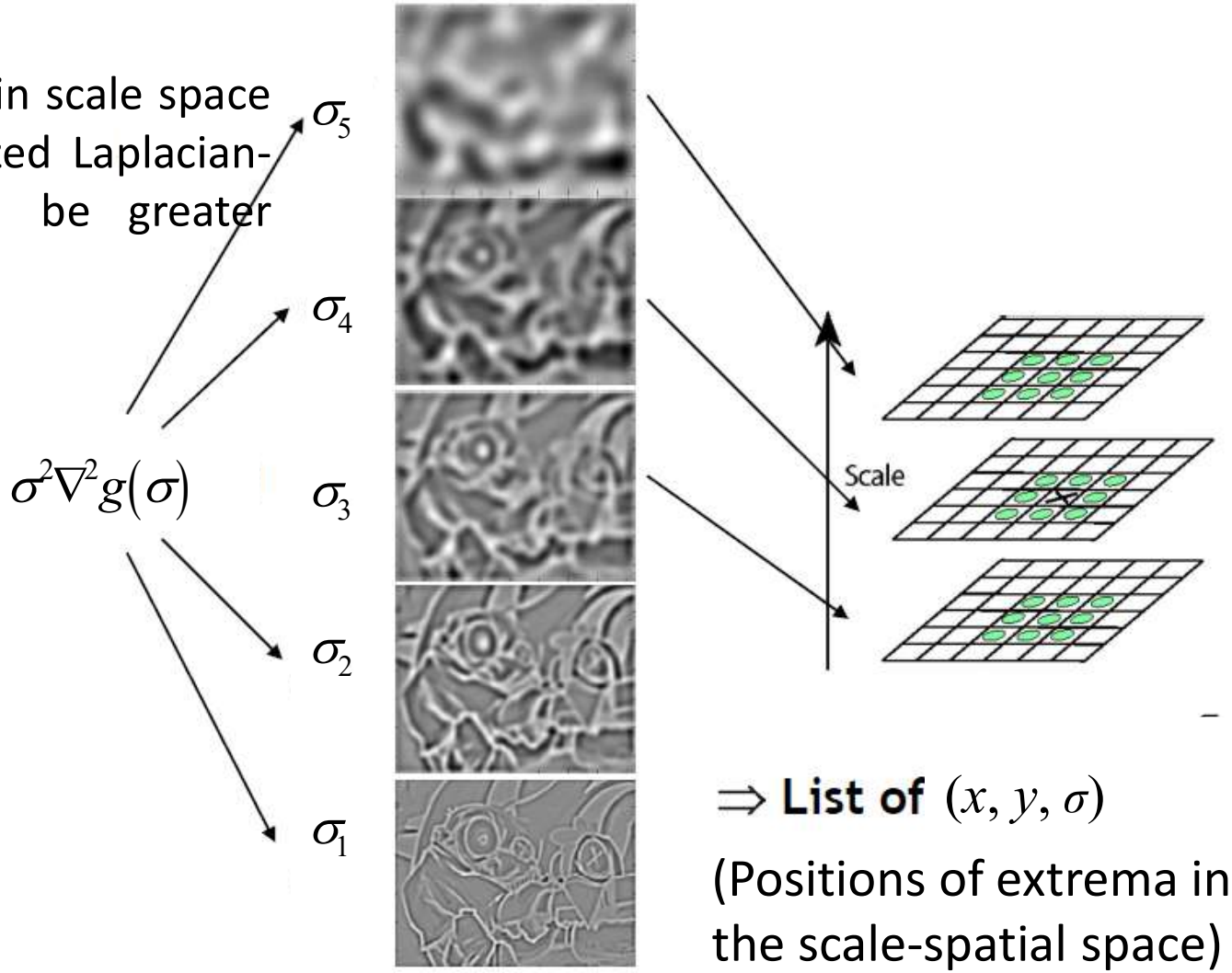




# SIFT feature—point detector (implementation)

- Interest points:
  - Local extremum in scale space of scale-normalized Laplacian-of-Gaussian and be greater than a threshold






$\Rightarrow$ **List of** $(x, y, \sigma)$

(Positions of extrema in the scale-spatial space)

# SIFT feature—point detector (implementation)

Note: local extrema is obtained as key points by comparing the examined location with all the other 26 points around it in the scale-space

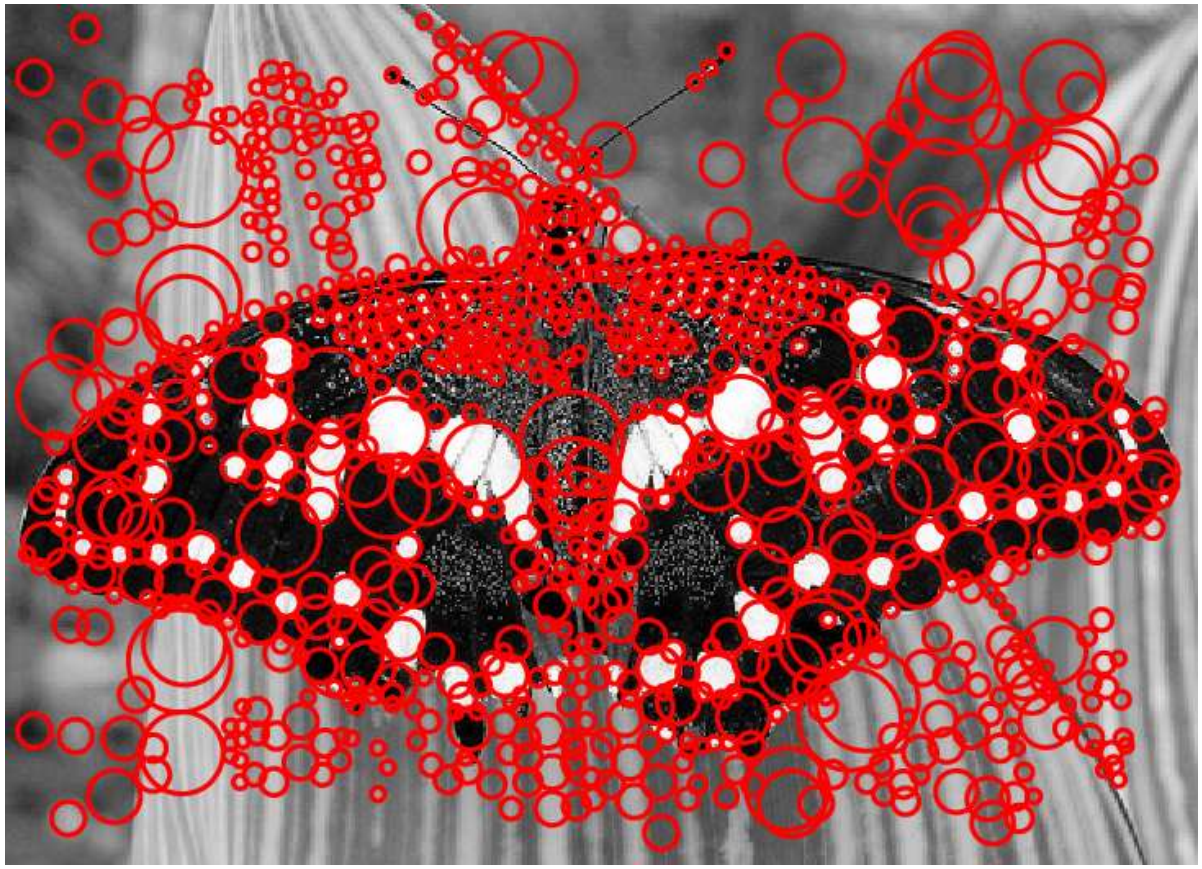
If the local extrema of scale-normalized LoG is achieved at $\boldsymbol{p}$, two things of $\boldsymbol{p}$ can be determined: its spatial location and characteristic scale

# SIFT feature—point detector (example)

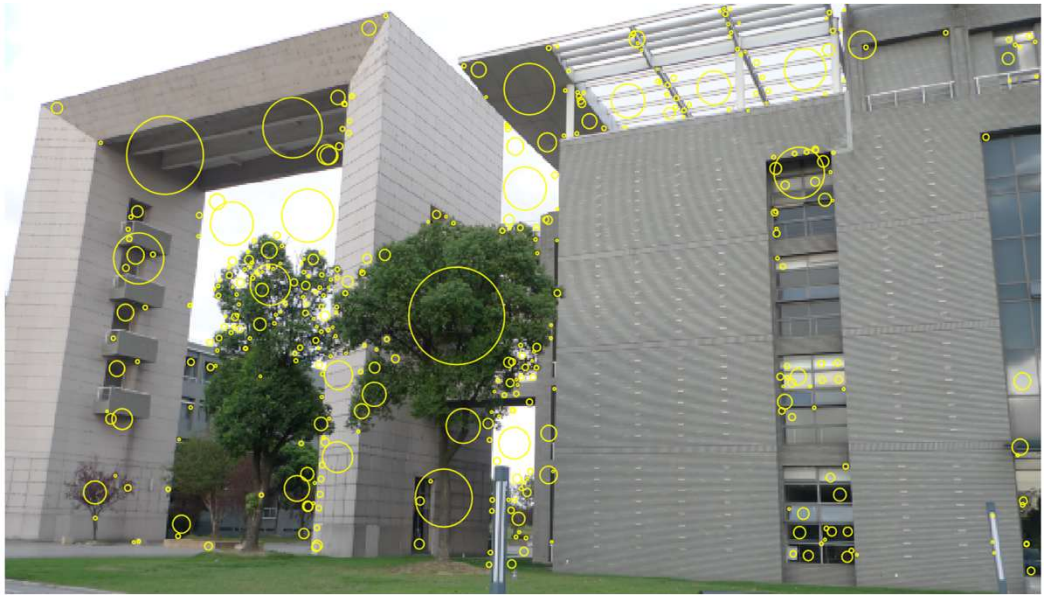
# SIFT feature—point detector (example)

sigma = 11.9912

# SIFT feature—point detector (example)



# SIFT feature—point detector (example)



# Content

- General requirements for interest point detectors and descriptors
- Harris feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
- SIFT feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
- ORB feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
  - Multiscale processing
- Feature point matching
- Practice

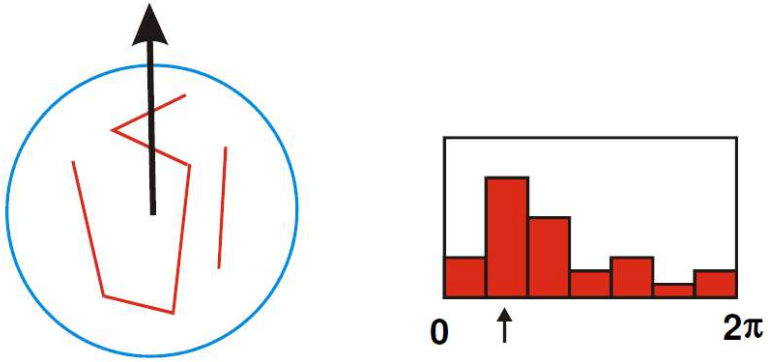
# SIFT feature—feature descriptor

- Building the SIFT (scale invariant feature transform) descriptor for the key-point $\boldsymbol{p}$, whose characteristic scale is $\sigma$
  - Take a $c\sigma \times c\sigma$ region $\mathcal{R}$ around $\boldsymbol{p}$

# SIFT feature—feature descriptor

- Building the SIFT (scale invariant feature transform) descriptor for the key-point $\boldsymbol{p}$, whose characteristic scale is $\sigma$
  - Take a $c\sigma \times c\sigma$ region $\mathcal{R}$ around $\boldsymbol{p}$
  - Compute the dominant orientation of the region $\mathcal{R}$
    - Compute the histogram of the gradient orientation of all the points in $\mathcal{R}$
    - The gradient orientation falls in the range $[0, 2\pi)$
    - Such a histogram usually has 36 bins; the voting of a pixel $\boldsymbol{p}_i$ to the assigned bin (by $\boldsymbol{p}_i$'s gradient orientation) is $\omega_i \cdot m_i$

      where $\omega_i$ is a Gaussian weight according to the distance between $\boldsymbol{p}_i$ and $\boldsymbol{p}$, and $m_i$ is $\boldsymbol{p}_i$'s gradient magnitude

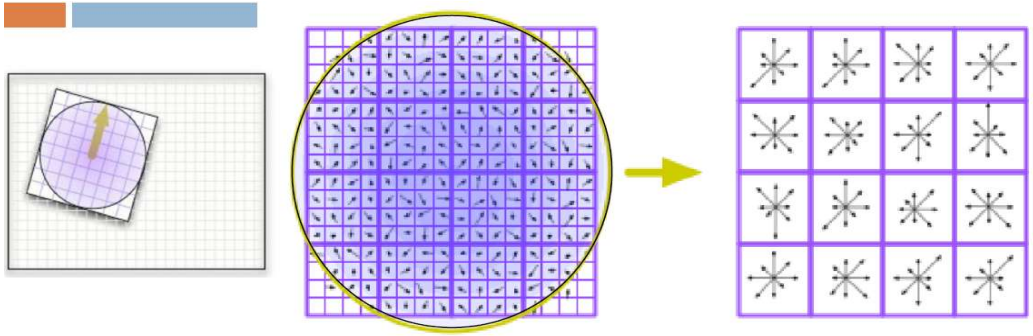


# SIFT feature—feature descriptor

- Building the SIFT (scale invariant feature transform) descriptor for the key-point $\boldsymbol{p}$, whose characteristic scale is $\sigma$
  - Take a $c\sigma \times c\sigma$ region $\mathcal{R}$ around $\boldsymbol{p}$
  - Compute the dominant orientation of the region $\mathcal{R}$
  - Rotate the gradients and coordinates in $\mathcal{R}$ by the previously computed dominant orientation; this can achieve rotation invariant description

# SIFT feature—feature descriptor

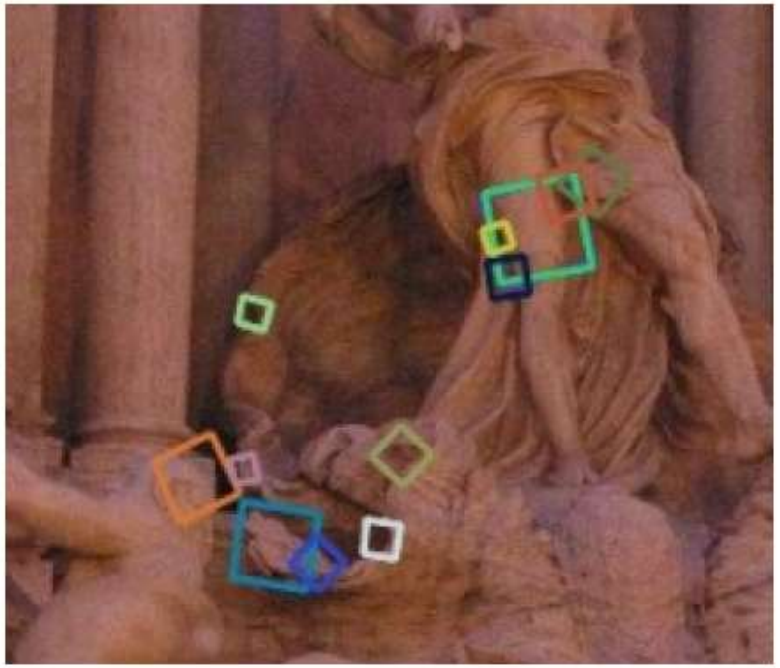
- Building the SIFT (scale invariant feature transform) descriptor for the key-point $\boldsymbol{p}$, whose characteristic scale is $\sigma$
  - Take a $c\sigma \times c\sigma$ region $\mathcal{R}$ around $\boldsymbol{p}$
  - Compute the dominant orientation of the region $\mathcal{R}$
  - Rotate the gradients and coordinates in $\mathcal{R}$ by the previously computed dominant orientation; this can achieve rotation invariant description
  - Separate the rotated region $\mathcal{R}$ to $4\times4$ sub-regions
  - Create gradient-orientation histogram for each sub-region with 8 bins (In real implementation, each sample point is weighted by a Gaussian weight according to its distance to $\boldsymbol{p}$)




The final SIFT descriptor for the point p is a $128d$ vector

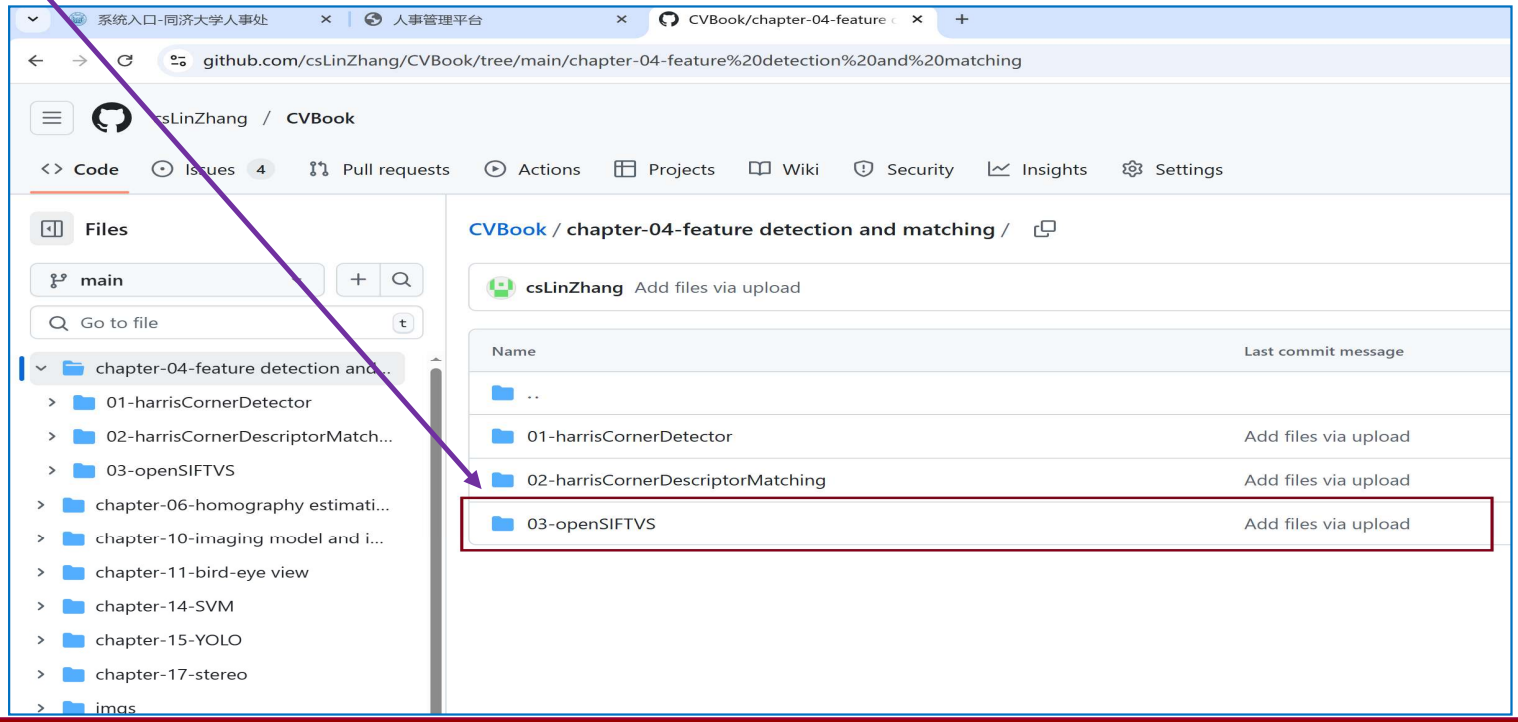
# SIFT feature—feature descriptor

- **One image yields:**
  - $n$ 128-dimensional descriptors: each one is a histogram of the gradient orientations within a patch
    - [n x 128 matrix]
  - $n$ scale parameters specifying the size of each patch
    - [n x 1 vector]
  - $n$ orientation parameters specifying the angle of the patch
    - [n x 1 vector]
  - $n$ 2D points giving positions of the patches
    - [n x 2 matrix]



# SIFT feature—feature descriptor

Note: in the lecture, we simply introduce the general ideas of SIFT. For more efficient and accurate implementation, Lowe suggested many tricks, such as using DoG to approximate LoG, performing detection in the DoG pyramid, refining key point locations, filtering out edge points, refining the domiant orientation etc. If you have interests in these details, you can refer to the textbook and the reference implementation



# Content

- General requirements for interest point detectors and descriptors
- Harris feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
- SIFT feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
- ORB feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
  - Multiscale processing
- Feature point matching
- Practice

# SIFT feature—distance between two descriptors

Given two SIFT descriptors $\boldsymbol{d}_1, \boldsymbol{d}_2 \in \mathbb{R}^n$ , their distance can be computed in different ways

(1) Sum of squared differences (SSD): $SSD_{dist}\left(\boldsymbol{d}_1, \boldsymbol{d}_2\right) = \left\|\boldsymbol{d}_1 - \boldsymbol{d}_2\right\|_2^2 = \sum_{i=1}^{n}\left(d_1^i - d_2^i\right)^2$

(2) Sum of absolute differences (SAD): $SAD_{dist}\left(\boldsymbol{d}_1, \boldsymbol{d}_2\right) = \sum_{i=1}^{n}\left|d_1^i - d_2^i\right|$

(3) Normalized cross correlation (NCC):

$$NCC_{dist}\left(\boldsymbol{d}_1, \boldsymbol{d}_2\right) = 1 - \frac{1}{n}\frac{\left(\boldsymbol{d}_1 - \mu\left(\boldsymbol{d}_1\right)\right)\cdot\left(\boldsymbol{d}_2 - \mu\left(\boldsymbol{d}_2\right)\right)}{std\left(\boldsymbol{d}_1\right)std\left(\boldsymbol{d}_2\right)}$$

or

$$NCC_{dist}\left(\boldsymbol{d}_1, \boldsymbol{d}_2\right) = \arccos\left(\frac{1}{n}\frac{\left(\boldsymbol{d}_1 - \mu\left(\boldsymbol{d}_1\right)\right)\cdot\left(\boldsymbol{d}_2 - \mu\left(\boldsymbol{d}_2\right)\right)}{std\left(\boldsymbol{d}_1\right)std\left(\boldsymbol{d}_2\right)}\right)$$

where $\mu\left(\boldsymbol{d}_1\right)$ returns the mean value of $\boldsymbol{d}_1$ and $std(\boldsymbol{d}_1)$ returns the standard deviation of $\boldsymbol{d}_1$

# Content

- General requirements for interest point detectors and descriptors
- Harris feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
- SIFT feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
- ORB feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
  - Multiscale processing
- Feature point matching
- Practice

# ORB feature—overview

- Properties of the SIFT feature
  - Pros: It is very robust and has a good scale-invariant capability
  - Cons: Its computational complexity is high and thus is not implementation efficient

⬇

ORB (Oriented FAST and Rotated BRIEF) feature is a good alternative

⬇

- Overview of the ORB feature
  - ✓ It improves the FAST (feature from accelerated segment test) scheme to detect key points, by computing the dominant orientation
  - ✓ It uses BRIEF (binary robust independent elementary feature) as the descriptor

# ORB feature—overview

## ORB

RUBLEE E, RABAUD V, KONOLOGE K, et al. ORB: An efficient alternative to SIFT or SURF[C]//Proc. IEEE Int'l. Conf. Computer Vision, 2011: 2564–2571.

## FAST

ROSTEN E, DRUMMOND T. Machine learning for high-speed corner detection[C]//Proc. European Conf. Computer Vision, 2006: 430-443.

## BRIEF

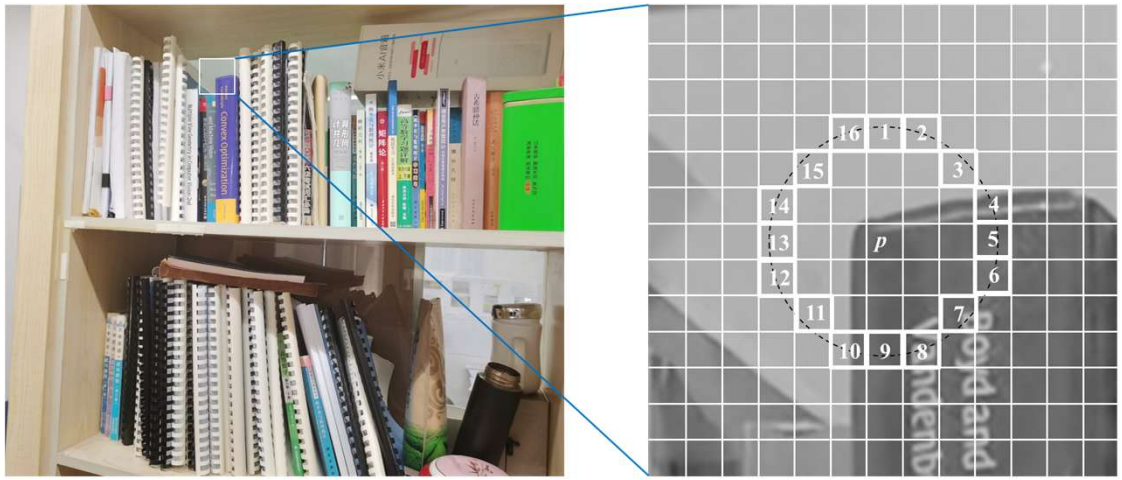
CALONDER M, LEPETIT V, STRECHA C, et al. BRIEF: Binary robust independent elementary features[C]//Proc. European Conf. Computer Vision, 2010: 778-792.

# ORB feature—point detector

- ORB is based on FAST corner
- FAST corner
  - Core idea is: if a pixel differs significantly from its neighboring pixels (being much brighter or darker), it is likely to be a corner point
  - Compared with other corner detection algorithms, FAST only needs to compare the brightness of pixels, making it very fast

# ORB feature—point detector

- FAST corner detection
  - Given a grayscale image $\boldsymbol{I}$, the process of determining whether a point $\boldsymbol{p}$ on it is a FAST feature point is as follows:
    - ✓ Set a threshold $t$ (typically set to 20% of $\boldsymbol{I}(\boldsymbol{p})$)
    - ✓ Take point $\boldsymbol{p}$ as the center and select 16 pixel positions on a circle with a radius of 3
    - ✓ If there are $N$ consecutive points on the selected circle whose brightness is greater than $\boldsymbol{I}(\boldsymbol{p})+t$ or less than $\boldsymbol{I}(\boldsymbol{p})-t$, then point $\boldsymbol{p}$ is considered a feature point; $N$ is usually taken as 12, and the corresponding feature point detection algorithm is called FAST-12




In the FAST-12 algorithm, a preprocessing test operation can be added to quickly exclude pixel positions that cannot be feature points. Specifically, for each pixel $\boldsymbol{p}$, directly test the pixel values at the 1st, 5th, 9th, and 13th positions on its neighborhood circle. Point $\boldsymbol{p}$ is only potentially a feature point if at least 3 of these 4 pixel values are simultaneously greater than $\boldsymbol{I}(\boldsymbol{p})+t$ or less than $\boldsymbol{I}(\boldsymbol{p})-t$; otherwise, $\boldsymbol{p}$ can be directly excluded

# ORB feature—point detector

- FAST corner detection
  - Given a grayscale image $\boldsymbol{I}$, the process of determining whether a point $\boldsymbol{p}$ on it is a FAST feature point is as follows:
    - ✓ Set a threshold $t$ (typically set to 20% of $\boldsymbol{I}(\boldsymbol{p})$)
    - ✓ Take point $\boldsymbol{p}$ as the center and select 16 pixel positions on a circle with a radius of 3
    - ✓ If there are $N$ consecutive points on the selected circle whose brightness is greater than $\boldsymbol{I}(\boldsymbol{p})+t$ or less than $\boldsymbol{I}(\boldsymbol{p})-t$, then point $\boldsymbol{p}$ is considered a feature point; $N$ is usually taken as 12, and the corresponding feature point detection algorithm is called FAST-12
    - ✓ After obtaining the preliminary FAST feature points, similar to the Harris corner detection, non-maximum suppression is required to retain only the points with maximum response values within a certain region, thereby obtaining reasonable sparse feature points; The response value of a FAST feature point can be calculated as the average of the differences between the pixel values of the $N$ consecutive points and $\boldsymbol{I}(\boldsymbol{p})$

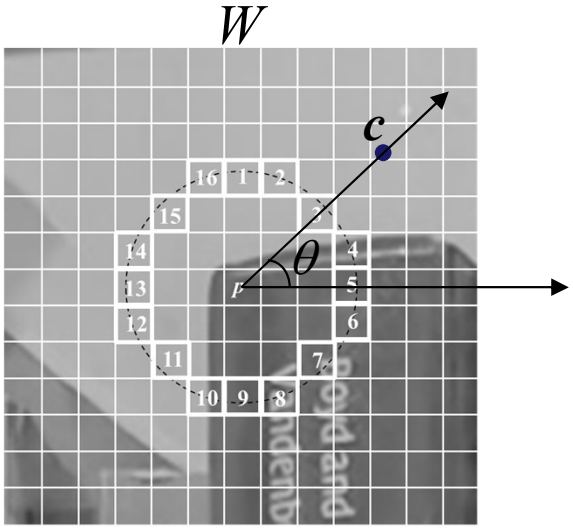
# ORB feature—point detector

- FAST corner detection
- Dominant orientation for the FAST corner structure
  - To ensure the constructed feature descriptors are rotation-invariant, we need to extract the local dominant orientation information of feature points
  - Suppose $\boldsymbol{p}$ is a FAST feature point on a grayscale image $\boldsymbol{I}$. The local dominant orientation of point $\boldsymbol{p}$ can be computed using the intensity centroid method as follows:
    - ✓ Take a window $W$ centered at $\boldsymbol{p}$ and regard $\boldsymbol{p}$ as the origin of the local coordinate system
    - ✓ Define the moments of the image patch within $W$,

$$m_{pq} = \sum_{(x,y)\in W} x^p y^q W(x,y)$$

    - ✓ The centroid $\boldsymbol{c}$ of the image patch $W$ is $\boldsymbol{c} = \left( \frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right)$
    - ✓ The local dominant orientation of $\boldsymbol{p}$ is,

$$\theta = \arctan2(m_{01}, m_{10}),\ \theta \in [0, 2\pi)$$

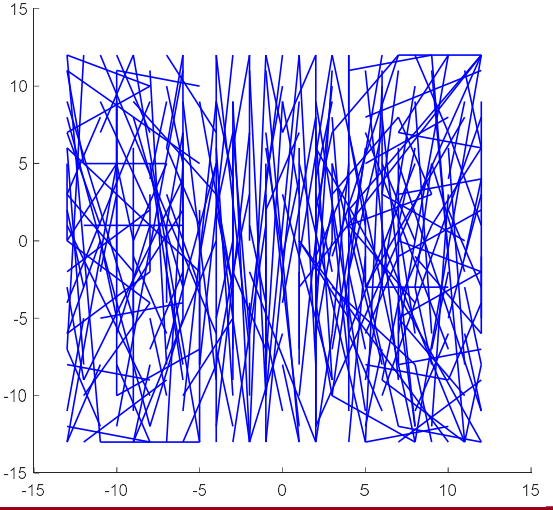


# Content

- General requirements for interest point detectors and descriptors
- Harris feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
- SIFT feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
- ORB feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
  - Multiscale processing
- Feature point matching
- Practice

# ORB feature—feature descriptor

- FAST uses BRIEF descriptor
  - BRIEF is a binary descriptor, consisting of 0s and 1s; 0 and 1 encode the magnitude relationship between pixel values at two random positions (e.g., $a$ and $b$) within an $s \times s$ (where $s$ is typically $31$) window centered at the key point: if $a$ is greater than $b$, the value is 1; otherwise, it is 0
  - By selecting 128 groups (or 256, 512 groups) of random positions for magnitude comparison encoding, a $128d$ (or 256, 512-dimensional) vector composed of 0s and 1s, such as BRIEF-128 (or BRIEF-256, BRIEF-512), can be obtained; of course, in the same application, the random pattern of 128 groups (or 256, 512 groups) of position pairs must be pre-fixed




The random position selection pattern of the BRIEF descriptor used in OpenCV. It selects a total of 256 groups of positions for pixel value comparison.

# ORB feature—feature descriptor

- FAST uses BRIEF descriptor
  - BRIEF is a binary descriptor, consisting of 0s and 1s; 0 and 1 encode the magnitude relationship between pixel values at two random positions (e.g., $a$ and $b$) within an $s \times s$ (where $s$ is typically $31$) window centered at the key point: if $a$ is greater than $b$, the value is 1; otherwise, it is 0
  - By selecting 128 groups (or 256, 512 groups) of random positions for magnitude comparison encoding, a $128d$ (or 256, 512-dimensional) vector composed of 0s and 1s, such as BRIEF-128 (or BRIEF-256, BRIEF-512), can be obtained; of course, in the same application, the random pattern of 128 groups (or 256, 512 groups) of position pairs must be pre-fixed
  - To make the BRIEF descriptor rotation-invariant: For a feature point $\boldsymbol{p}$, assuming its dominant orientation is $\theta$, when computing the BRIEF vector for $\boldsymbol{p}$, each pre-selected sampling position coordinate used for encoding is first rotated by $\theta$ around $\boldsymbol{p}$. Subsequently, the pixel values at the newly obtained positions after rotation are employed in the subsequent encoding process. This approach ensures that the ultimately derived BRIEF vector exhibits rotational invariance

# Content

- General requirements for interest point detectors and descriptors
- Harris feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
- SIFT feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
- ORB feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
  - Multiscale processing
- Feature point matching
- Practice

# ORB feature—distance between two descriptors

- Since the BRIEF feature vector is represented as a binary string composed of 0s and 1s, the **(normalized) Hamming distance** is more suitable for calculating the distance between two BRIEF vectors compared to the previously introduced $SSD_{dist}$, $SAD_{dist}$, and $NCC_{dist}$

Suppose $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are two binary BRIEF feature vectors. Their normalized Hamming distance can be calculated as follows,

$$H_{dist}\left(\boldsymbol{b}_1, \boldsymbol{b}_2\right) = \frac{ones\_num\left(\boldsymbol{b}_1 \odot \boldsymbol{b}_2\right)}{size\left(\boldsymbol{b}_1\right)}$$

where $\boldsymbol{b}_1 \odot \boldsymbol{b}_2$ denotes the bitwise XOR operation between $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$, $ones\_num(\boldsymbol{x})$ returns the number of 1s in the binary string $\boldsymbol{x}$, and $size(\boldsymbol{b}_1)$ returns the bit length of the binary string $\boldsymbol{b}_1$

# Content

- General requirements for interest point detectors and descriptors
- Harris feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
- SIFT feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
- ORB feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
  - Multiscale processing
- Feature point matching
- Practice

# ORB feature—Multiscale processing

- ORB features aforementioned exhibit rotational invariance and demonstrate robustness against illumination changes and noise; however, they lack scale invariance
- Suppose $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$ are two images of the same physical scene captured at different scales. To achieve feature point matching between them, we construct image pyramids $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively. FAST feature points are extracted and corresponding BRIEF feature descriptors are constructed at each level of $\mathcal{G}_1$ and $\mathcal{G}_2$ (using fixed hyperparameters for FAST detection and BRIEF descriptor construction across all levels). Finally, the positions of the feature points are converted to the resolution of the input images.
- For a given image $\boldsymbol{I}$, its image pyramid can be constructed as follows:
  - ✓ Let $scale\_factor$ be the scaling factor (typically set to $1.2$)
  - ✓ The scale parameter for the $l$-th level of the pyramid is $s_l = scale\_factor^{l\text{-}1}$ ($l = 1, 2, L$), where $L$ is the total number of pyramid levels
  - ✓ The $l$-th level image is obtained through interpolation-based downsampling, with a resolution $1/s_l$ of the original image $\boldsymbol{I}$

# ORB feature—Multiscale processing

……

Compare the differences in strategies used by SIFT and ORB for scale-invariant feature point matching

# Content

- General requirements for interest point detectors and descriptors
- Harris feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
- SIFT feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
- ORB feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
  - Multiscale processing
- Feature point matching
- Practice

# Feature point matching

Suppose $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$ are two images

$\boldsymbol{I}_1$'s key points and the associated descriptors are $\{\boldsymbol{x}_i\}_{i=1}^{m}$ and $\mathcal{P}=\{\boldsymbol{d}_i\}_{i=1}^{m}$

$\boldsymbol{I}_2$'s key points and the associated descriptors are $\{\boldsymbol{y}_j\}_{j=1}^{n}$ and $\mathcal{Q}=\{\boldsymbol{e}_j\}_{j=1}^{n}$

If and only if $\boldsymbol{x}_i$' and $\boldsymbol{y}_j$'s descriptors $\boldsymbol{d}_i$ and $\boldsymbol{e}_j$ satisfy the following conditions, we say the key points $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$ match and form a correspondence pair,

1) $dist(\boldsymbol{d}_i, \boldsymbol{e}_j)<t_1$, where $t_1$ is a predefined threshold

2) $\boldsymbol{d}_i$ and $\boldsymbol{e}_j$ satisfy the "two-direction" confirmation criteria, i.e.,

$$\forall \boldsymbol{v}\in\mathcal{P}, \boldsymbol{v}\neq\boldsymbol{d}_i, dist\left(\boldsymbol{v},\boldsymbol{e}_j\right)>dist\left(\boldsymbol{d}_i,\boldsymbol{e}_j\right)$$

$$\forall \boldsymbol{v}\in\mathcal{Q}, \boldsymbol{v}\neq\boldsymbol{e}_j, dist\left(\boldsymbol{d}_i,\boldsymbol{v}\right)>dist\left(\boldsymbol{d}_i,\boldsymbol{e}_j\right)$$

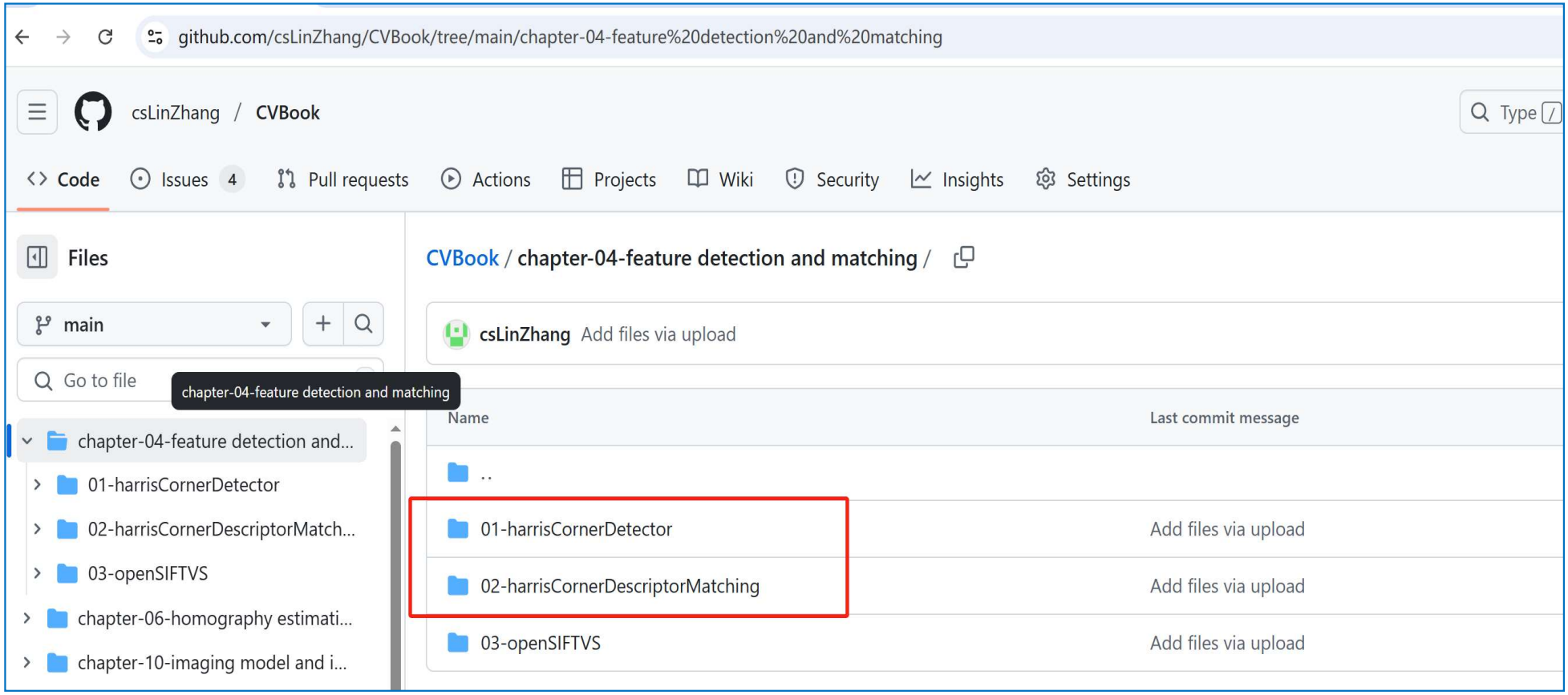
3) $\boldsymbol{d}_i$ and $\boldsymbol{e}_j$'s matching is unambiguous

Let $d_1=dist\left(\mathbf{d}_i,\mathbf{e}_j\right)$. Suppose $\boldsymbol{e}_k$ is the second best matching descriptor to $\boldsymbol{d}_i$ and let $d_2=dist\left(\boldsymbol{d}_i,\boldsymbol{e}_k\right)$. Then, $d_1/d_2<t_2$, where $t_2$ is another predefined threshold

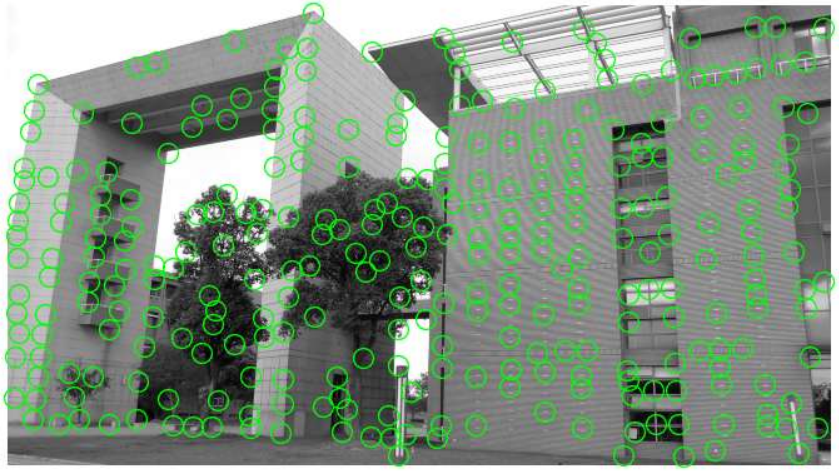
# Content

- General requirements for interest point detectors and descriptors
- Harris feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
- SIFT feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
- ORB feature
  - Point detector
  - Feature descriptor
  - Distance between two descriptors
  - Multiscale processing
- Feature point matching
- Practice

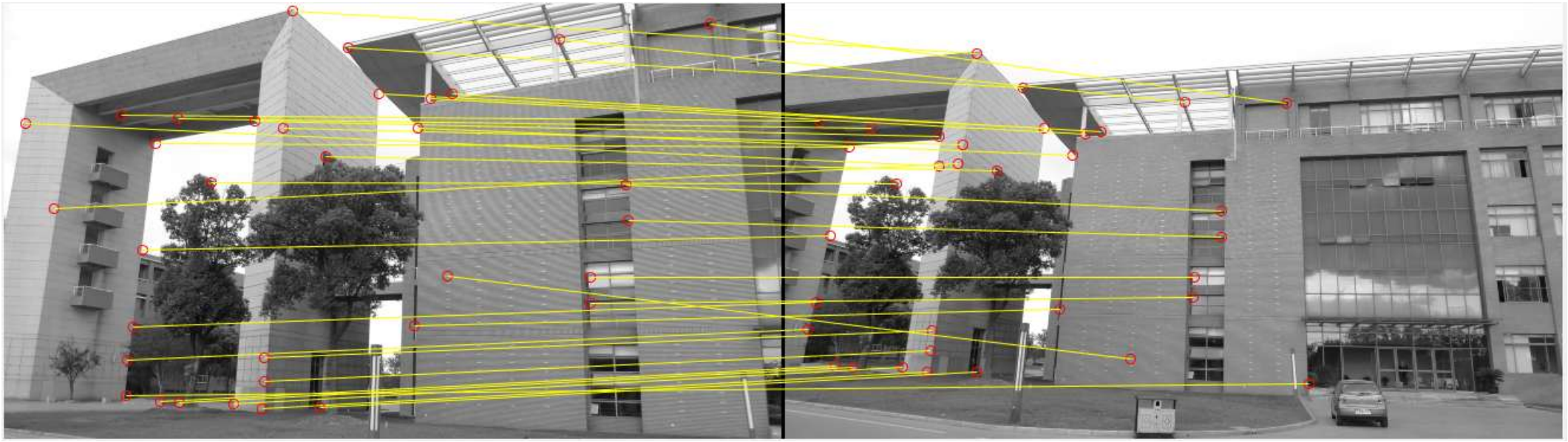
# Practice—Feature matching with Harris features



# Practice—Feature matching with Harris features

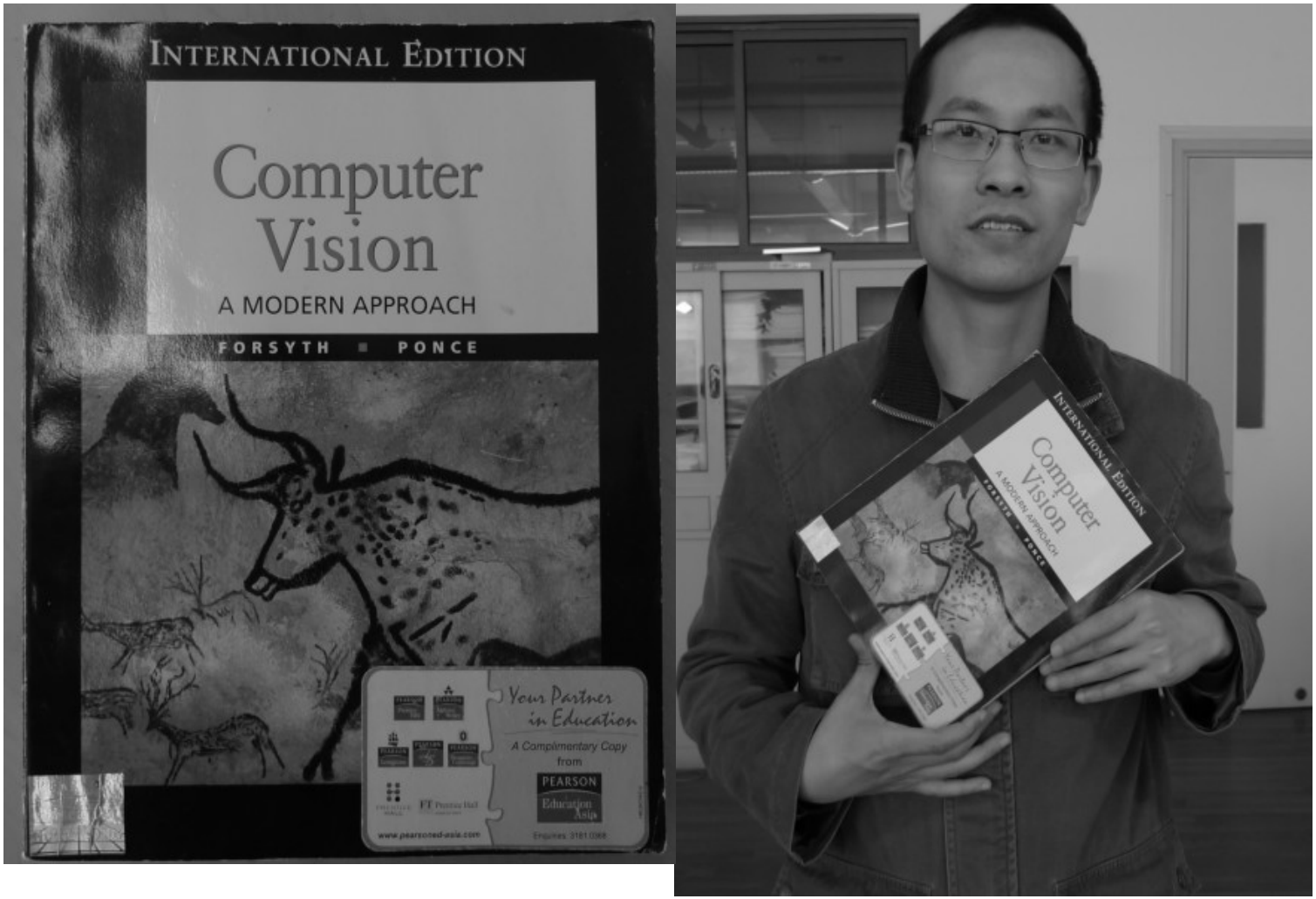


# Practice—Feature matching with Harris features



Matching Harris corners using block descriptors

# Practice—Feature matching with SIFT features



# Practice—Feature matching with SIFT features


vlfeat.org

**VLFeat.org**

Home | Download | Tutorials | Applications | Documentation

Home

The **VLFeat** open source library implements popular computer vision algorithms specializing in image understanding and local features extraction and matching. Algorithms include Fisher Vector, VLAD, SIFT, MSER, k-means, hierarchical k-means, agglomerative information bottleneck, SLIC superpixels, quick shift superpixels, large scale SVM training, and many others. It is written in C for efficiency and compatibility, with interfaces in MATLAB for ease of use, and detailed documentation throughout. It supports Windows, Mac OS X, and Linux. The latest version of VLFeat is 0.9.21.

ACM OpenSource Award

### Download
- **VLFeat 0.9.21** (Windows, Mac, Linux)
- Source code and installation
- git repository, bug tracking.

### Documentation
- MATLAB commands
- C API with algorithm descriptions
- Command line tools

### Tutorials
- Features: Covariant detectors, HOG, SIFT, MSER, Quick shift, SLIC
- Clustering: IKM, HIKM, AIB
- Matching: Randomized kd-trees
- All tutorials

### Example applications
- Caltech-101 classification
- SIFT matching for auto-stitching
- All example applications

### Citing
```
@misc{vedaldi08vlfeat,
 Author = {A. Vedaldi and B. Fulkerson},
 Title = {{VLFeat}: An Open and Portable Library
          of Computer Vision Algorithms},
 Year  = {2008},
 Howpublished = {\url{http://www.vlfeat.org/}}
}
```

### Acknowledgments
PASCAL2 Yandex UCLA Vision Lab Oxford VGG.

### News
*8/1/2018* **VLFeat 0.9.21 released**

Maintenance release. Fixed `vl_argparse` to be compatible with MatConvNet. Fixed the binaries for recent versions of macOS.

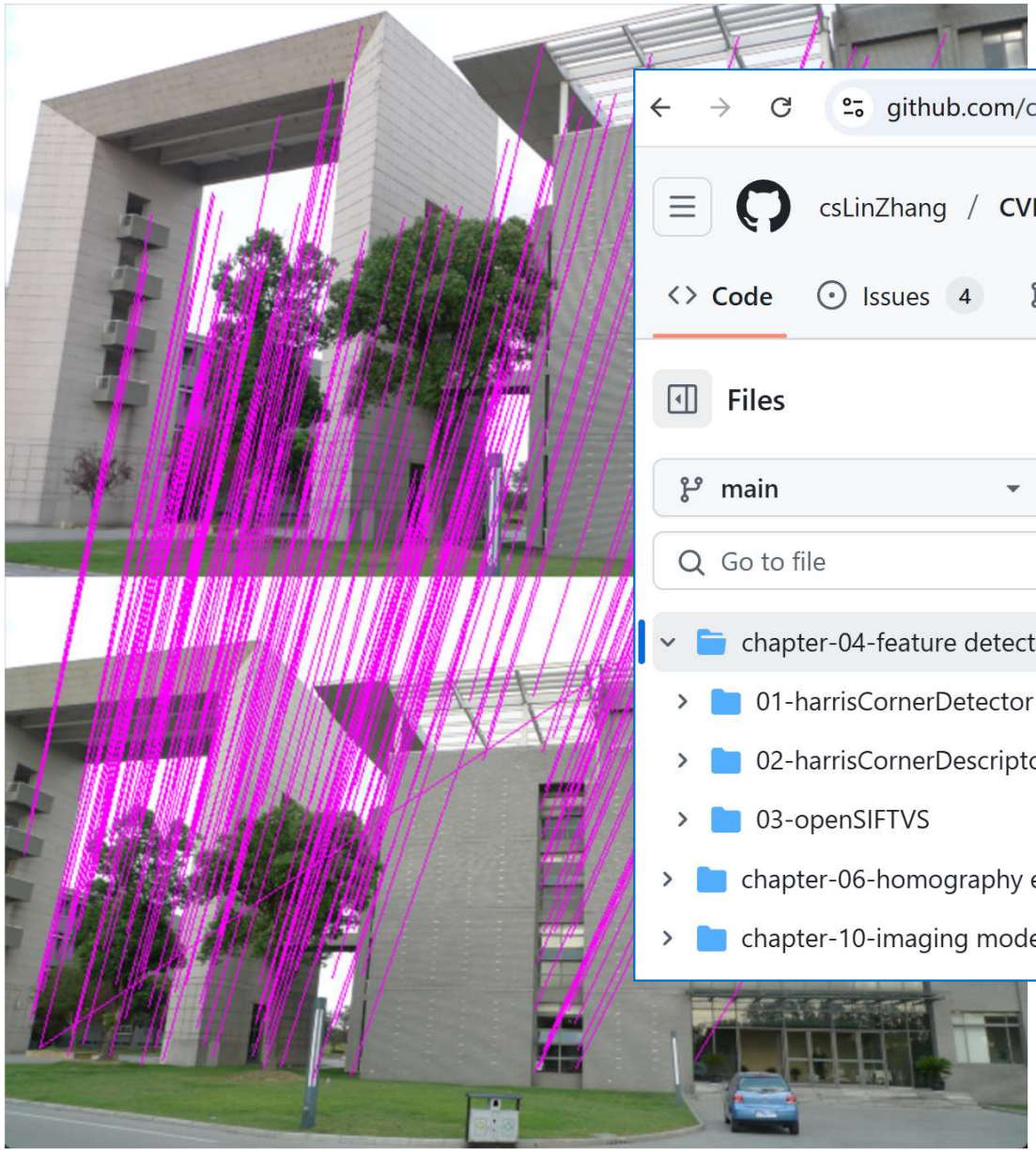
*14/1/2015* **VLFeat 0.9.20 released**

Maintenance release. Bugfixes.

*12/9/2014* ***MatConvNet***

Looking for an easy-to-use package to work with deep convolutional neural networks in MATLAB? Check out our new MatConvNet toolbox!

# Practice—Feature matching with SIFT features



github.com/csLinZhang/CVBook/tree/main/chapter-04-feature%20detection%20and%20matchin

csLinZhang / CVBook

Code | Issues 4 | Pull requests | Actions | Projects | Wiki | Security

Files

main

Go to file t

- chapter-04-feature detection and...
  - 01-harrisCornerDetector
  - 02-harrisCornerDescriptorMatch...
  - 03-openSIFTVS
- chapter-06-homography estimati...
- chapter-10-imaging model and i...

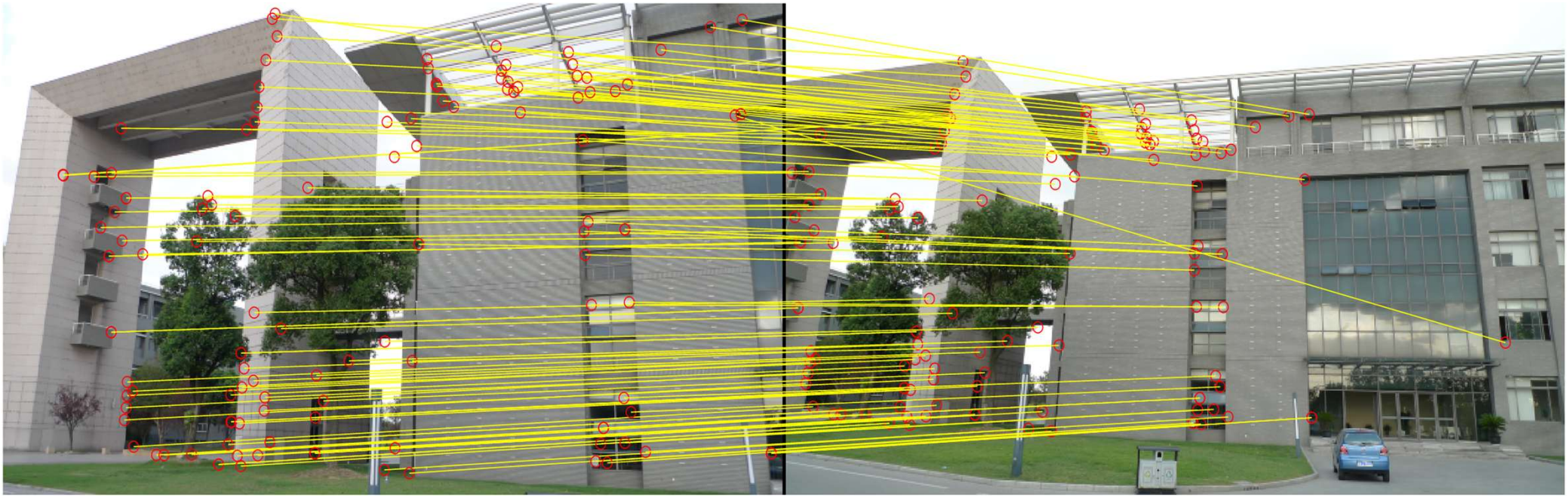
CVBook / chapter-04-feature detection and matching /

csLinZhang Add files via upload

Name

- ..
- 01-harrisCornerDetector
- 02-harrisCornerDescriptorMatching
- 03-openSIFTVS

# Practice—Feature matching with SIFT features



Matching scale-invariant points using SIFT descriptors (using Matlab library, for which details will be introduced in Chapter 6)

# Practice—Feature matching with SIFT features

如果我只想做一个“调包侠”……

Prompt to an LLM:

请帮我编写一段python程序，完成以下功能（可调用openCV库函数）：从硬盘中读入两张输入图像并命名为img1和img2；用SIFT特征点检测和描述子构建方法，分别从img1和img2上提取尺度不变特征点并构建相应描述子；基于描述子，对两张图像上的特征点进行匹配；将特征点匹配结果可视化出来，也就是把两张图像显示在一张大图上，用连线的方式显示特征点对应关系。


Q+A