

Handcrafted Feature Detection and Extraction

Pablo Mesejo

pmesejo@go.ugr.es

Universidad de Granada

Departamento de Ciencias de la Computación e Inteligencia Artificial



UNIVERSIDAD
DE GRANADA



DaSCI

Instituto Andaluz de Investigación en
Data Science and Computational Intelligence

Readings

- ***Harris corner detector***
 - Forsyth and Ponce (2012), Chapter 5.3.1
 - Szeliski (2022), Chapter 7.1
 - González and Woods (2018), Chapter 11.6
- ***Scale-Invariant Feature Transform (SIFT)***
 - Forsyth and Ponce (2012), Chapter 5.4.1
 - Szeliski (2022), Chapter 7.1
 - González and Woods (2018), Chapter 11.7
 - Otero & Delbracio (2014), “Anatomy of the SIFT Method”
- ***Histogram of Oriented Gradients (HOG)***
 - Forsyth and Ponce (2012), Chapter 5.4.2
 - Szeliski (2022), Chapter 6.3.2

Corner Detection

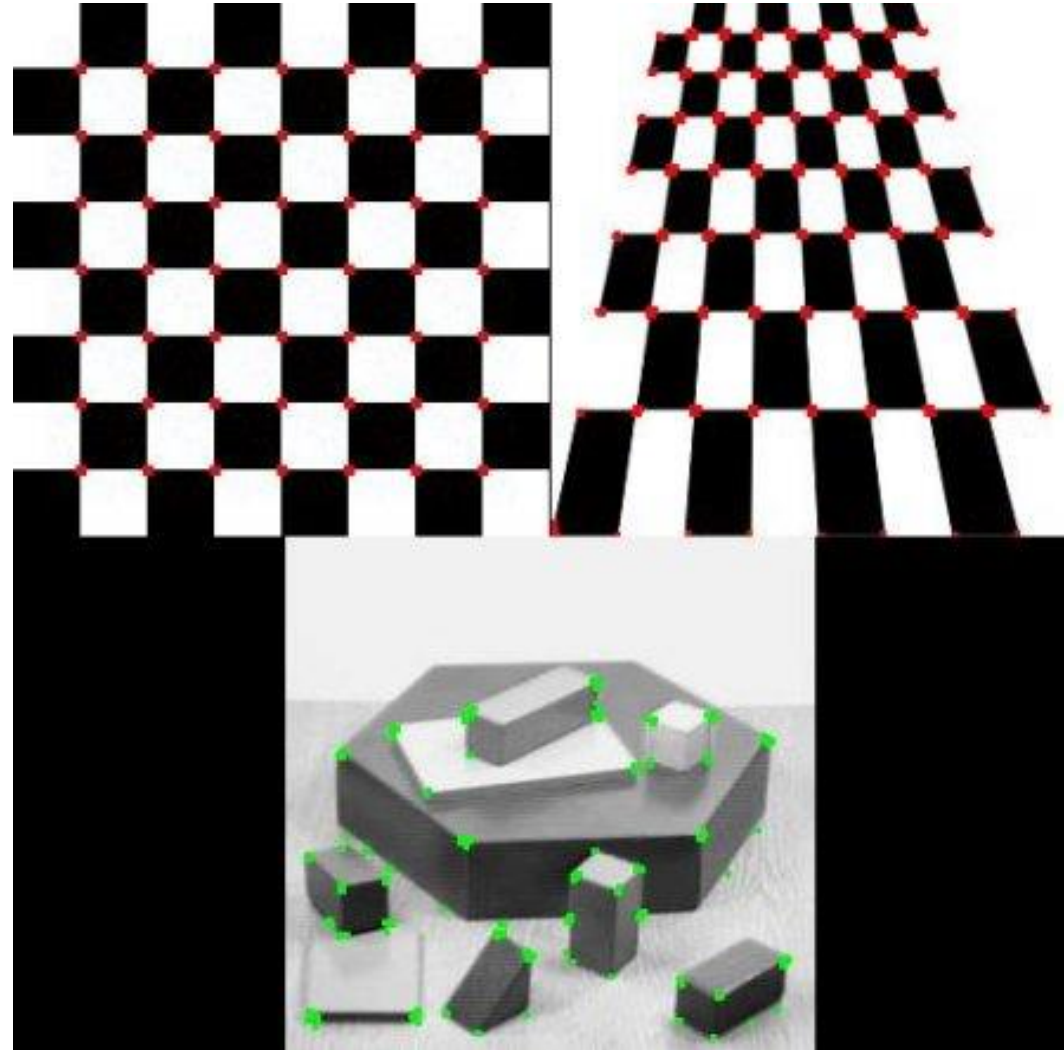
Harris Detector

We want to detect **corners**.

Why?

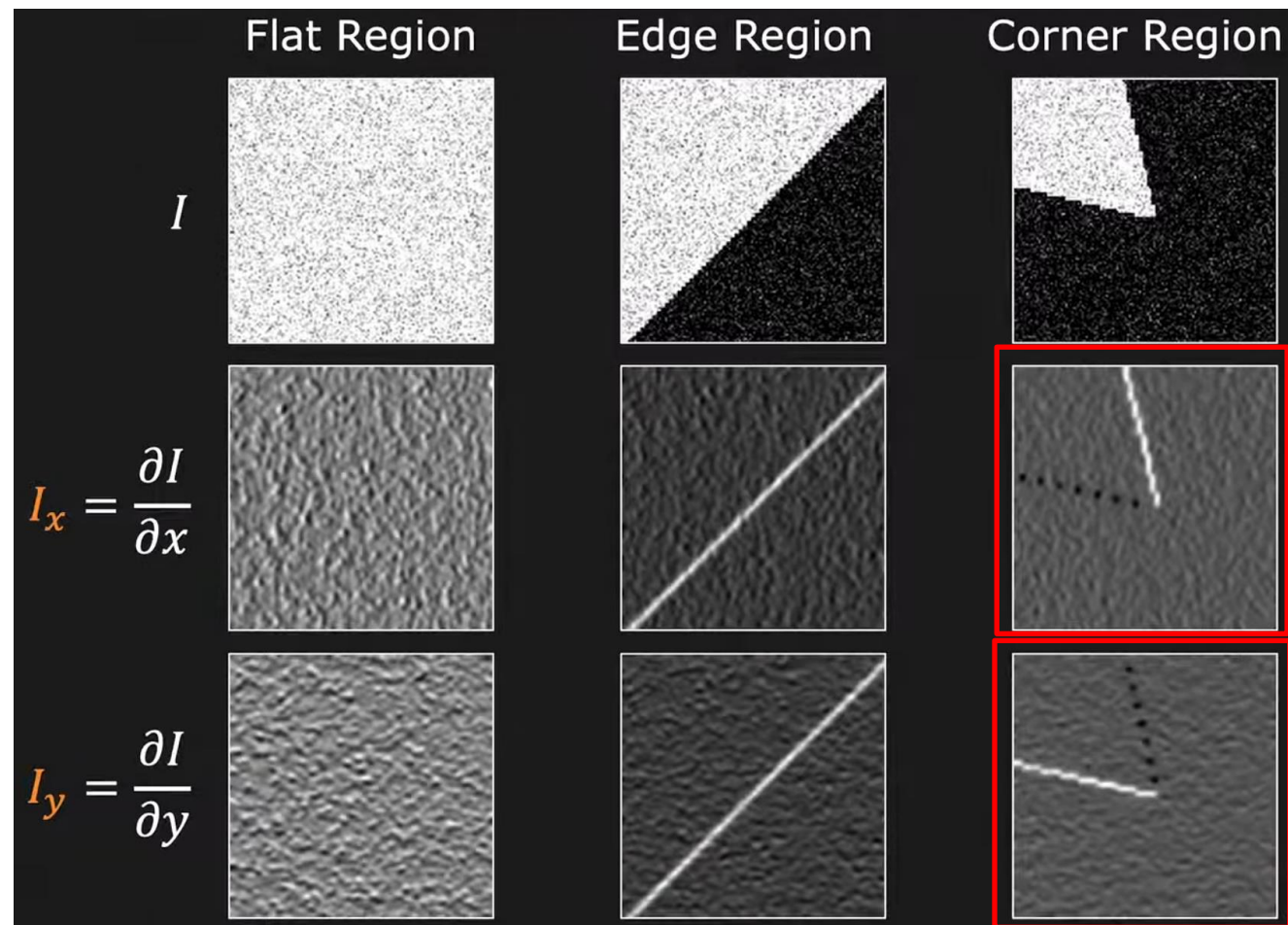
A corner can be localized reliably.

We look for image regions that are unusual/unique/distinctive, i.e., lead to unambiguous matches in other images.



Harris Detector

- We use image derivatives to find corners.

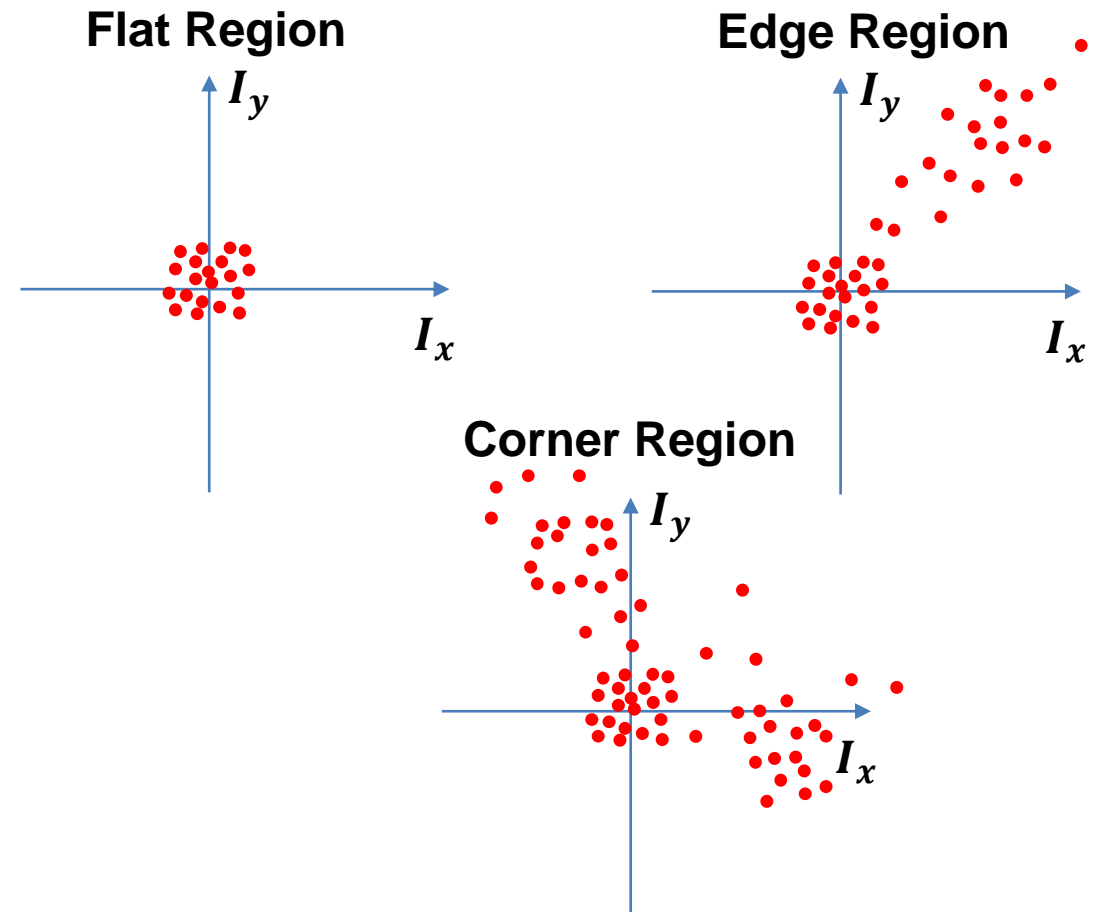
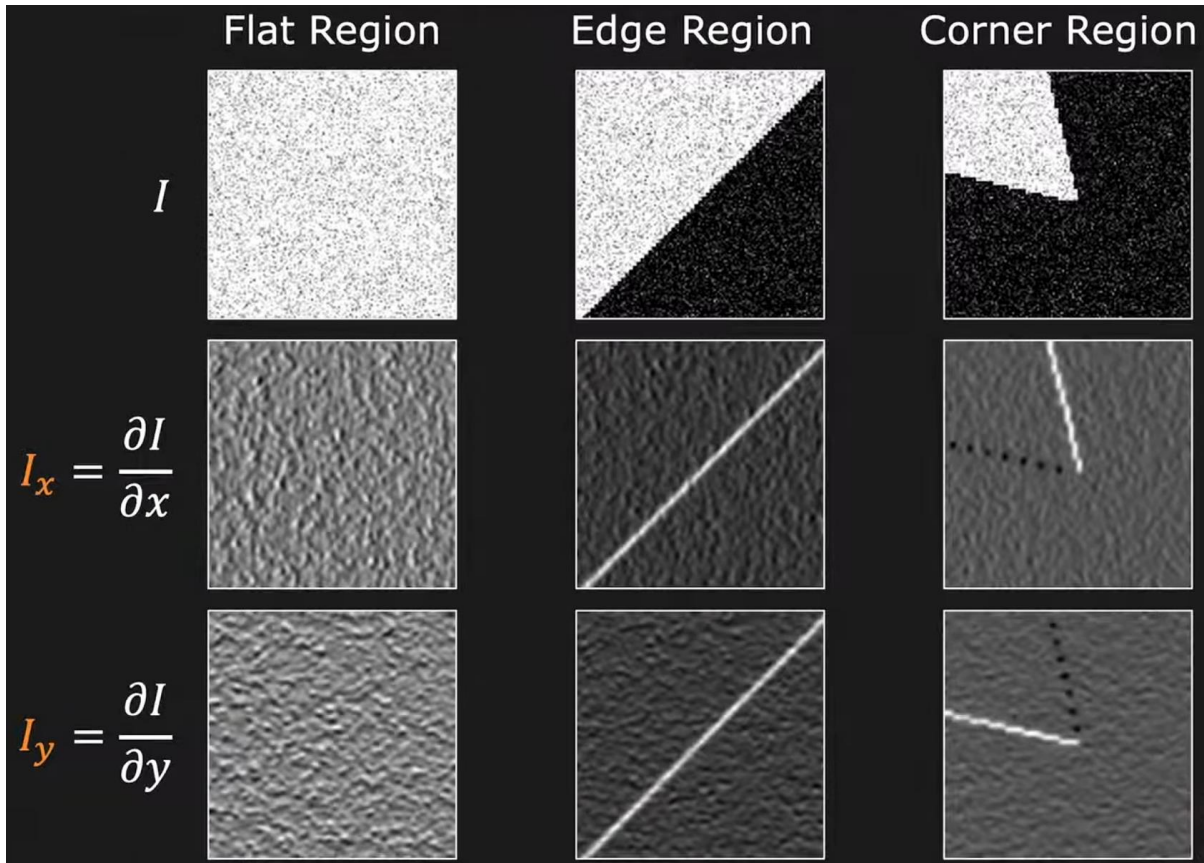


Reminder: Sobel Operator

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$$
$$\mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Harris Detector

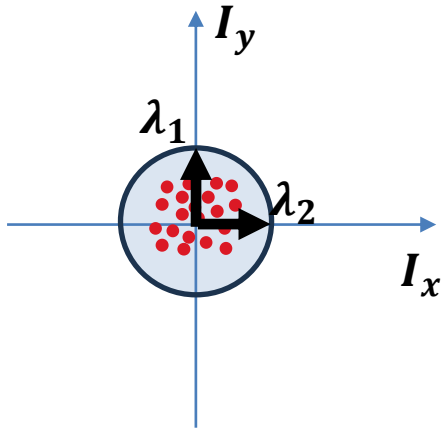
- Now, we are going to classify these three regions based on the distribution of the gradient within these regions.



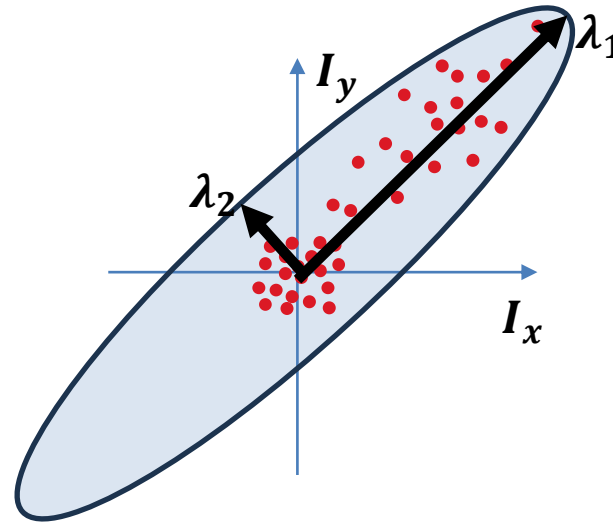
Harris Detector

- We fit an ellipse to the distribution (centered at the origin).

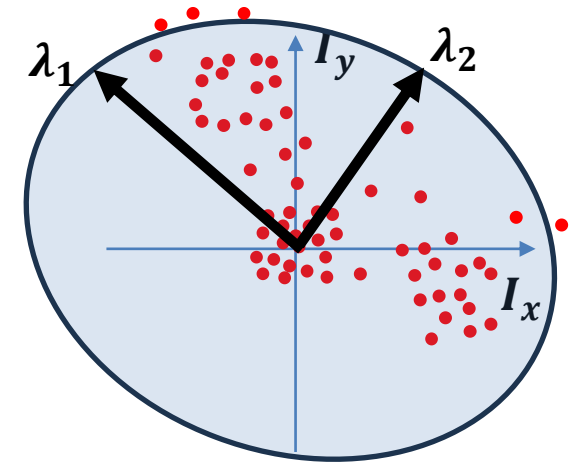
Flat Region



Edge Region



Corner Region



λ_1 : length of semi-major axis = maximum second moment = $\frac{1}{2} [a + c + \sqrt{b^2 + (a - c)^2}]$

λ_2 : length of semi-minor axis = minimum second moment = $\frac{1}{2} [a + c - \sqrt{b^2 + (a - c)^2}]$

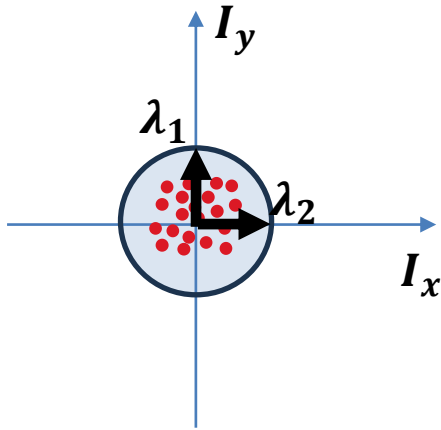
We'll see these expressions in more detail later!

$$a = \sum_{i \in W} (I_{x_i}^2) \quad c = \sum_{i \in W} (I_{y_i}^2) \quad b = 2 \sum_{i \in W} (I_{x_i} I_{y_i}) \quad W: \text{window centered at pixel}$$

Harris Detector

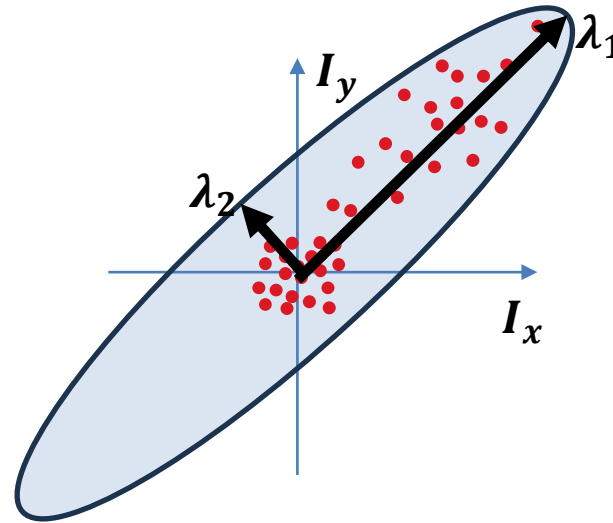
- We fit an ellipse to the distribution (centered at the origin).

Flat Region



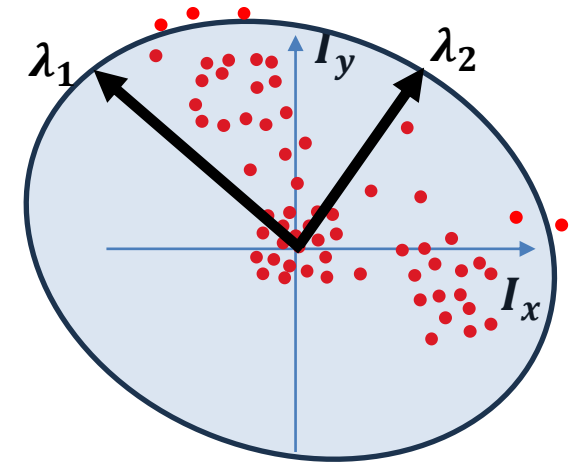
$\lambda_1 \approx \lambda_2$
Both are small

Edge Region



$\lambda_1 \gg \lambda_2$
 λ_1 is large
 λ_2 is small

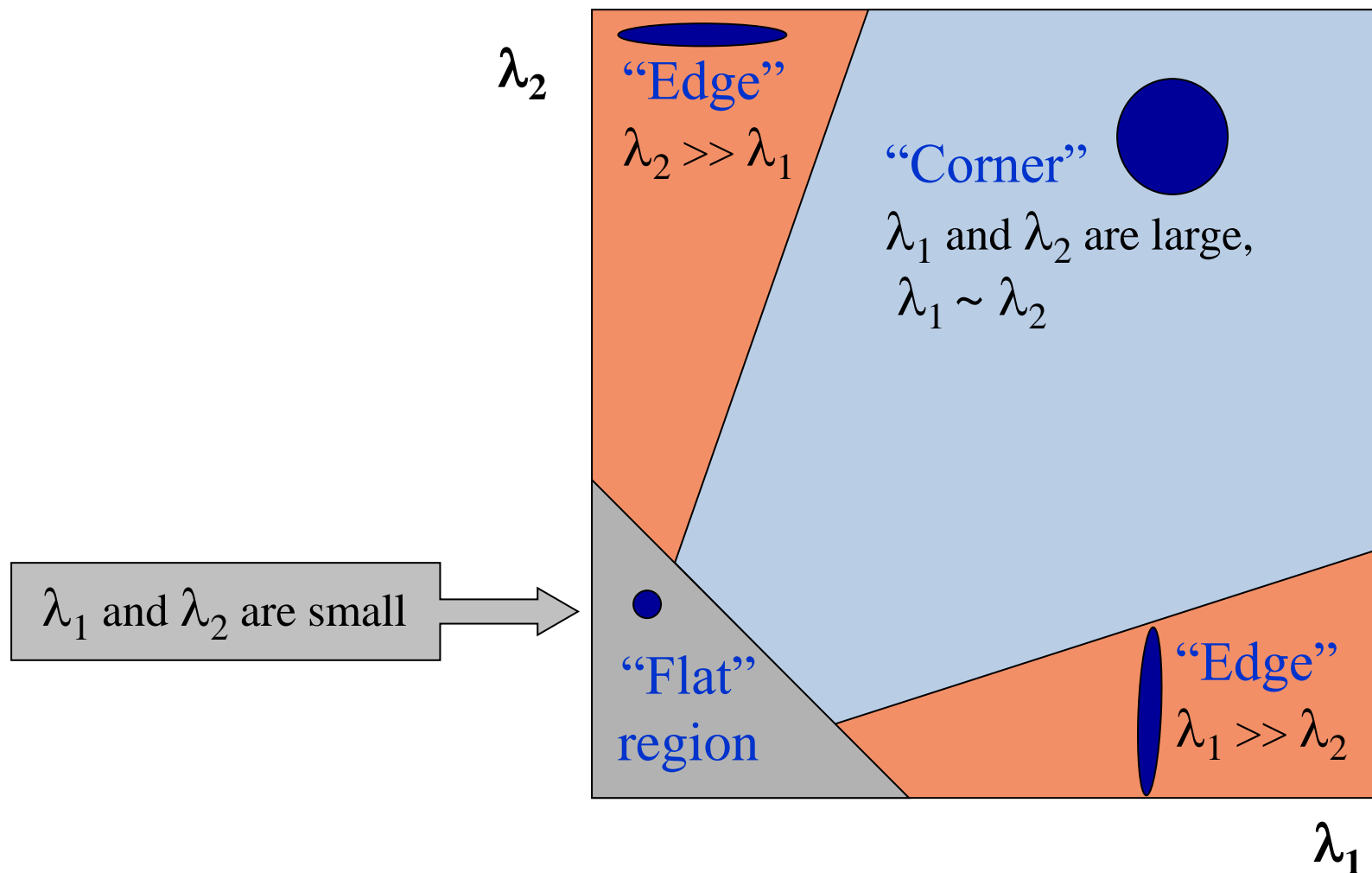
Corner Region



$\lambda_1 \approx \lambda_2$
Both are large

We classify our region just based on λ_1 and λ_2

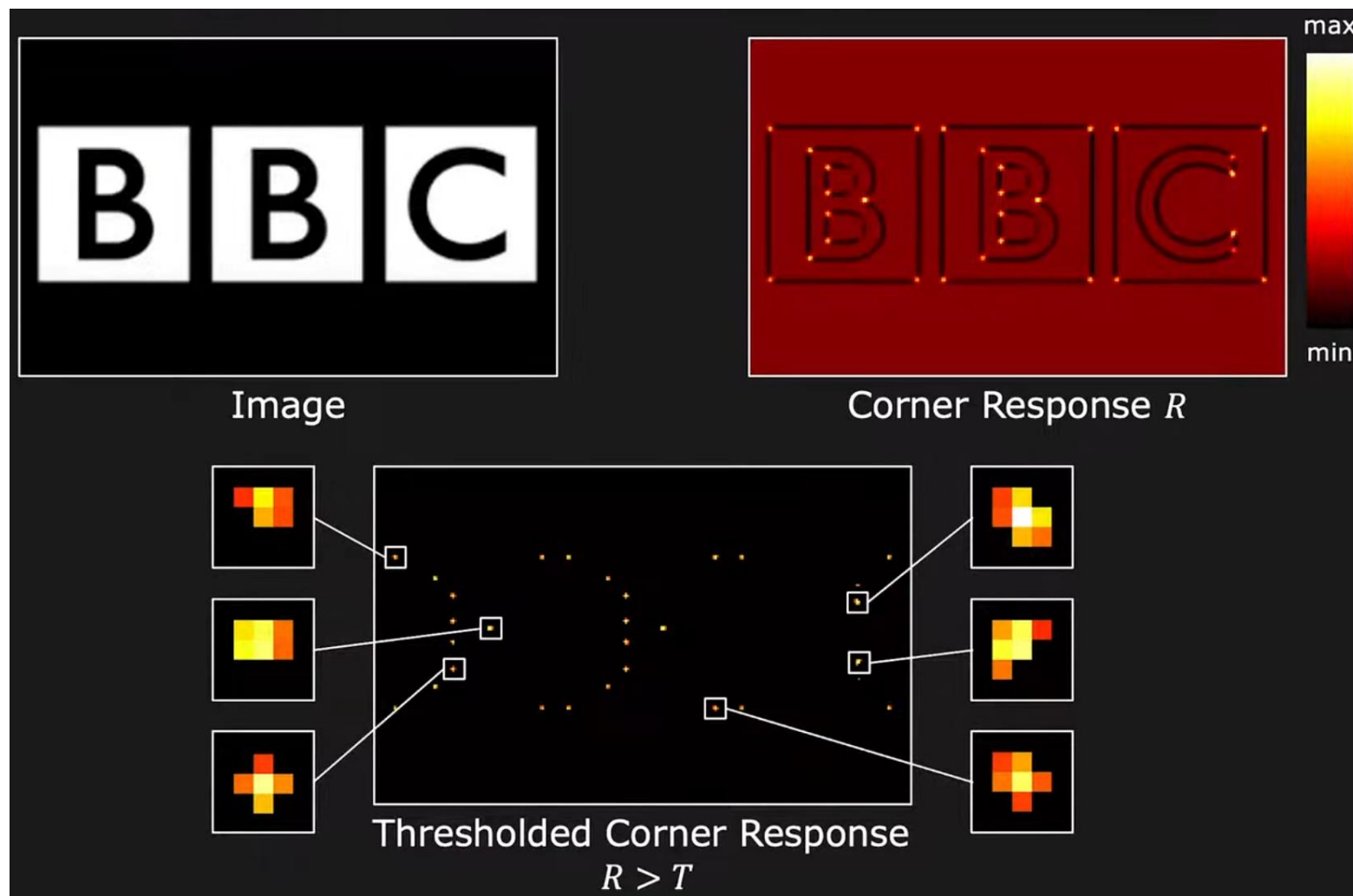
Harris Detector



$$R = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2$$

Where $0.04 \leq k \leq 0.06$ (designed empirically). Now, **when you apply a threshold to R , you can detect corners (interest points).**

Non-maxima suppression



How to determine the actual corner from these pixel clusters?

Non-maxima suppression

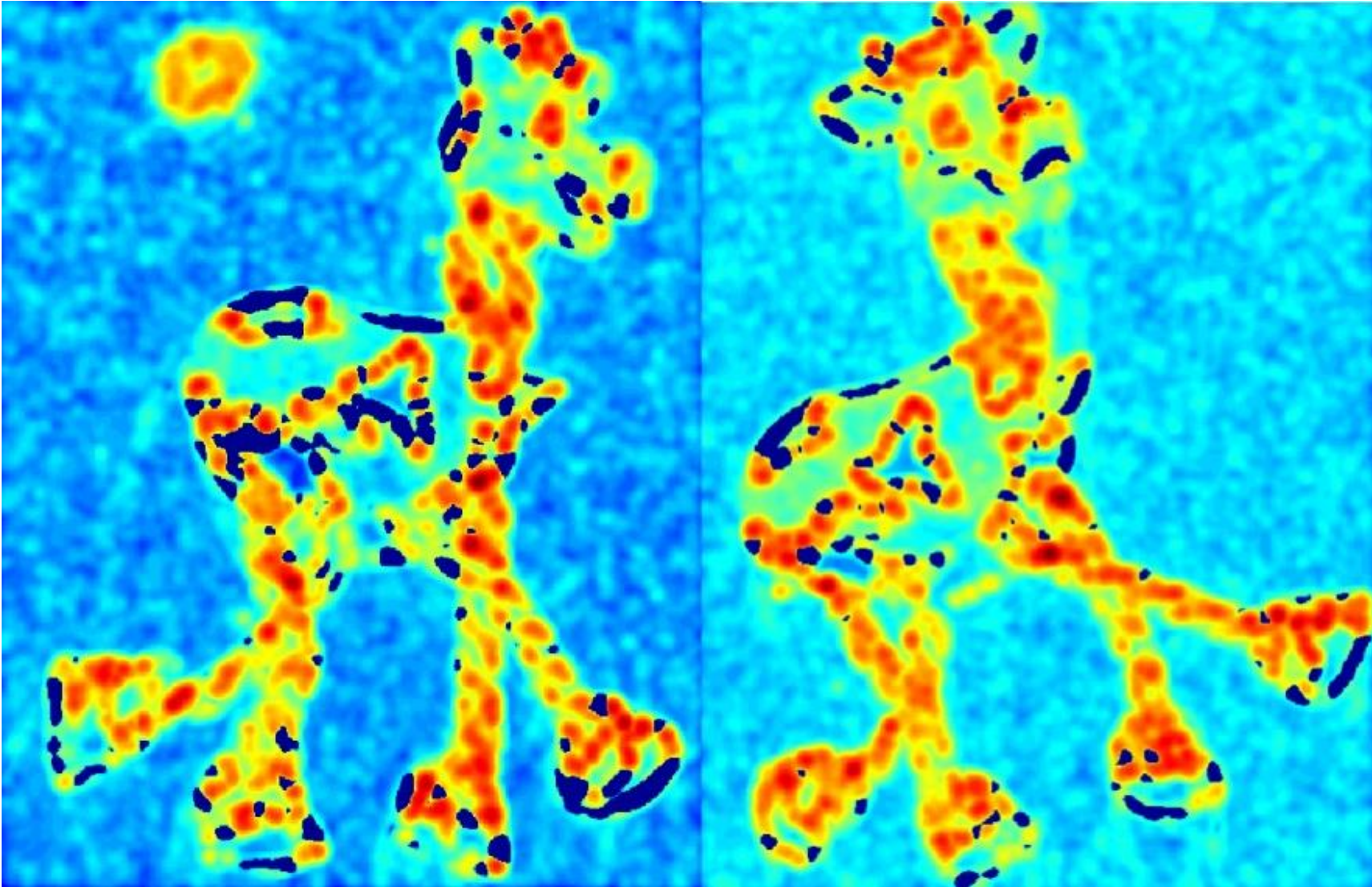
- 1) Slide a window of size k over the image
- 2) At each position, if the pixel at the center is the maximum value within the window, keep it. Otherwise, remove it.



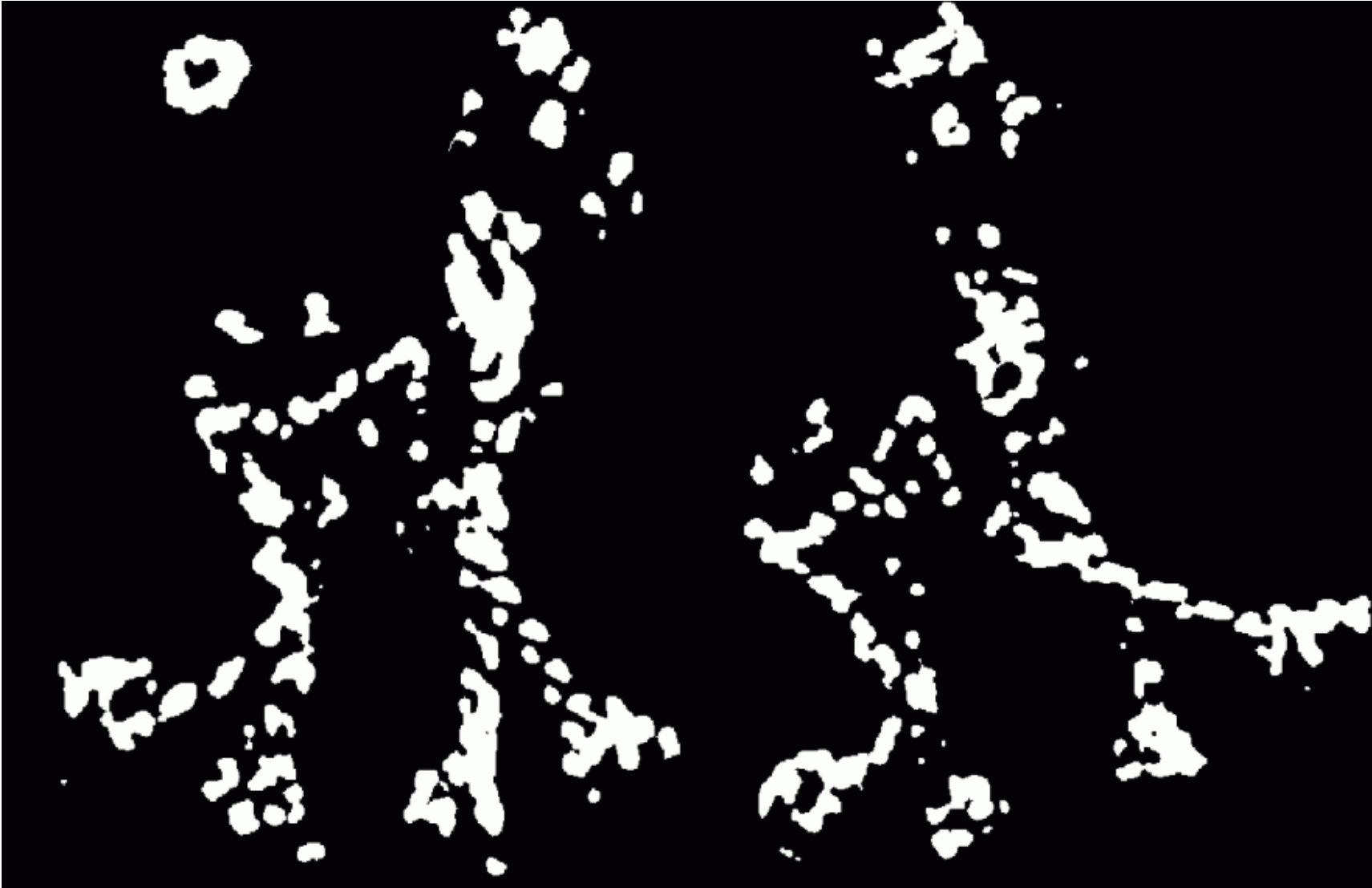
Harris detector example



Corner response value (red high, blue low)



Corner response thresholded



Find local maxima



Harris features (in red)



Harris Detector

- Let's now see the Harris corner detector from another (more mathematical) perspective

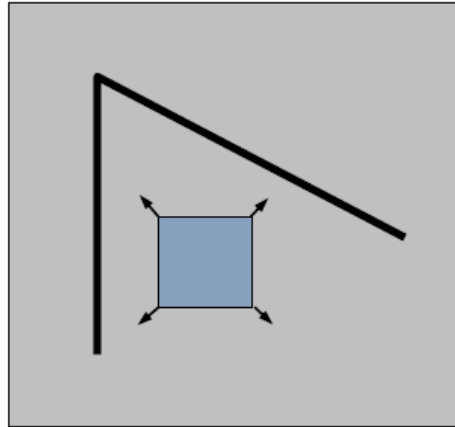
Next slides are mainly based on those by [Noah Snavely](#)

Harris Detector

Why are corners distinct?

Place a **small window** over a patch of constant image value.

If you slide the window in any direction, the image in the window will not change.



“flat” region:
no change in all
directions

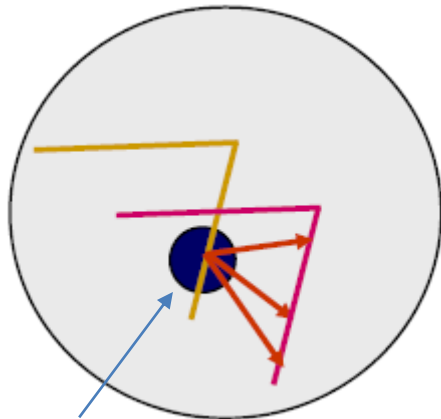
Harris Detector

Why are corners distinct?

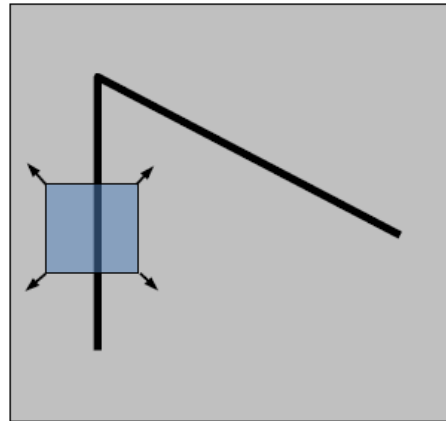
Place a **small window** over an edge.

If you **slide the window in the direction of the edge**, the image in the window will not change.

You cannot estimate location along an edge (a.k.a., aperture problem).



This feature detected in the yellow edge, where could be localized on the red one?



“edge”:
no change along the
edge direction

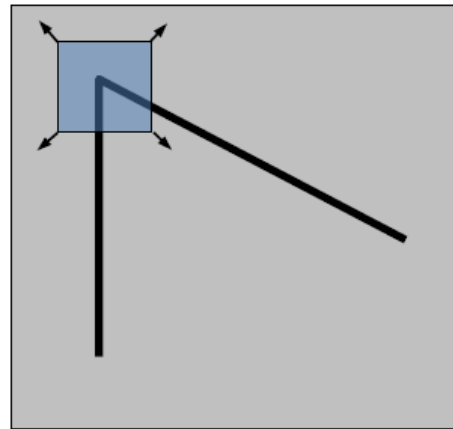
Harris Detector

Why are corners distinct?

Place a **small window** over a corner.

If you slide the window in any direction, the image in the window changes.

**Distinctive region. Wherever
I move, I detect changes.**



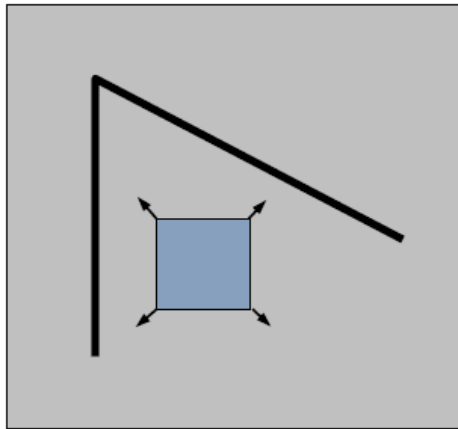
“corner”:
significant change in
all directions

Harris Detector

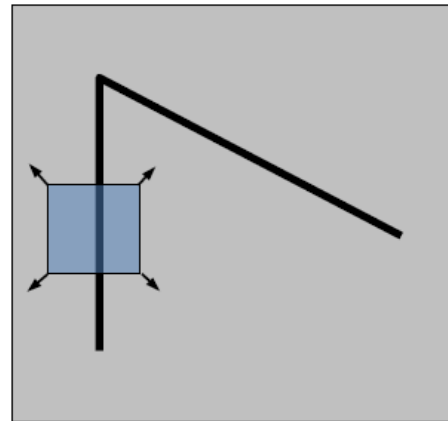
How do you find corners?

Easily recognized by looking through a small window.

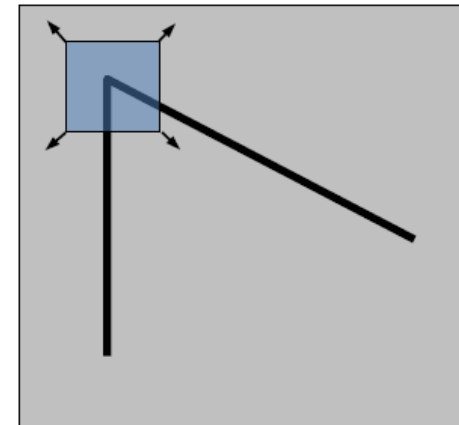
Shifting the window should give large change in intensity.



“flat” region:
no change in all
directions



“edge”:
no change along the
edge direction



“corner”:
significant change in
all directions

Harris Detector

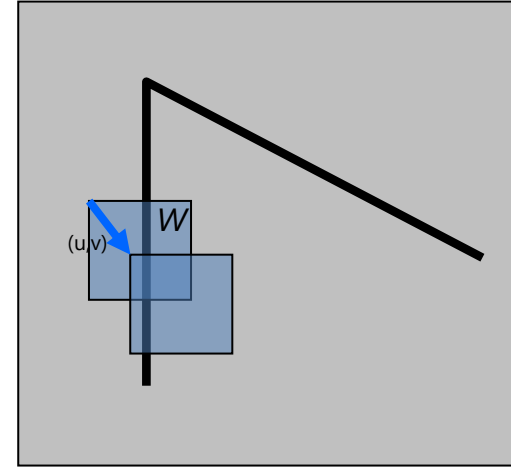
Consider shifting the window W by (u,v)

- how do the pixels in W change?
- compare each pixel before and after by summing up the squared differences (SSD)
- this defines an SSD "error" $E(u,v)$:

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

Annotations for the equation:

- $E(u, v)$: Error function
- (u, v) : Offset
- $I(x+u, y+v)$: Shifted Intensity
- $I(x, y)$: Intensity
- $(x, y) \in W$: We compute this SSD using all pixels of window W



- We are happy if this error is high
- We are very happy if this error is high *for all offsets*
 $(u,v) \rightarrow \text{possible corner!}$
- Slow to compute exactly for each pixel and each offset (u,v)

Corner detection: the math

Taylor Series expansion of I :

$$I(x+u, y+v) = I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

If the motion (u, v) is small (**small motion assumption**), then first order approximation is good

$$\begin{aligned} I(x+u, y+v) &\approx I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v \\ &\approx I(x, y) + [I_x \ I_y] \begin{bmatrix} u \\ v \end{bmatrix} \end{aligned}$$

shorthand: $I_x = \frac{\partial I}{\partial x}$



Plugging this into the formula on the previous slide...

Derivative of image I with respect to x-axis

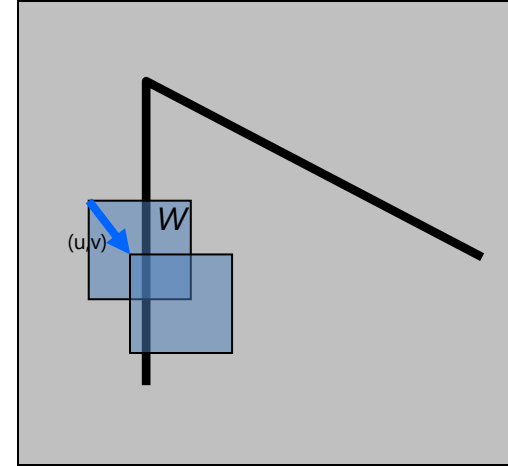
Note: if you have a function f , we can represent it in terms of the function and its derivatives at a point. These are the Taylor Series to approximate functions.

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

Corner detection: the math

Consider shifting the window W by (u, v)

- define an SSD "error" $E(u, v)$:

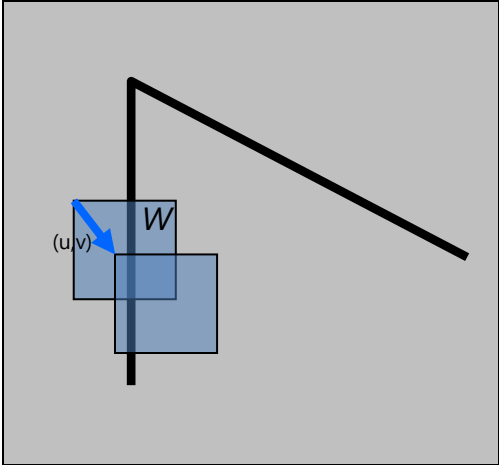


$$\begin{aligned} E(u, v) &= \sum_{(x, y) \in W} [I(x + u, y + v) - I(x, y)]^2 \\ &\approx \sum_{(x, y) \in W} [\cancel{I(x, y)} + I_x u + I_y v - \cancel{I(x, y)}]^2 \\ &\approx \sum_{(x, y) \in W} [I_x u + I_y v]^2 \end{aligned}$$

Corner detection: the math

Consider shifting the window W by (u, v)

- define an SSD "error" $E(u, v)$:

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

$$I_x^2 u^2 + I_y^2 v^2 + 2I_x u I_y v$$

$$A = \sum_{(x, y) \in W} I_x^2 \quad B = \sum_{(x, y) \in W} I_x I_y \quad C = \sum_{(x, y) \in W} I_y^2$$

- Thus, $E(u, v)$ is locally approximated as a quadratic error function

Corner detection: the math

$E(u, v)$ is locally approximated by a quadratic form.

$$E(u, v) \approx Au^2 + 2Buv + Cv^2$$

$$\approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_H \begin{bmatrix} u \\ v \end{bmatrix}$$

Note:

$$\begin{bmatrix} uA + vB & uB + vC \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = u^2A + uvB + uvB + v^2C$$

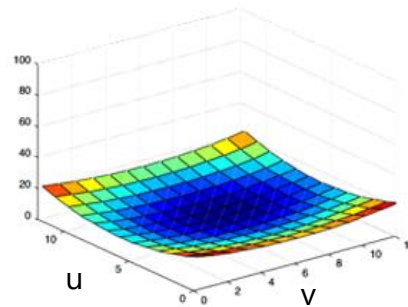
$$A = \sum_{(x,y) \in W} I_x^2$$

$$B = \sum_{(x,y) \in W} I_x I_y$$

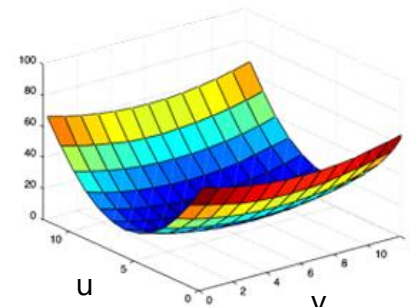
$$C = \sum_{(x,y) \in W} I_y^2$$

H

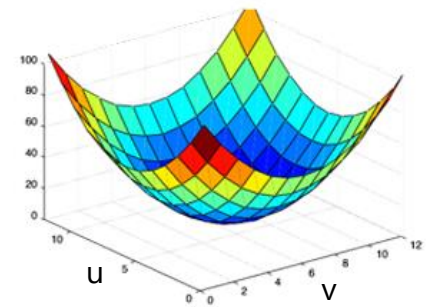
Image gradient covariance matrix (or 2nd moment matrix)



flat



edge
'line'



corner
'dot'

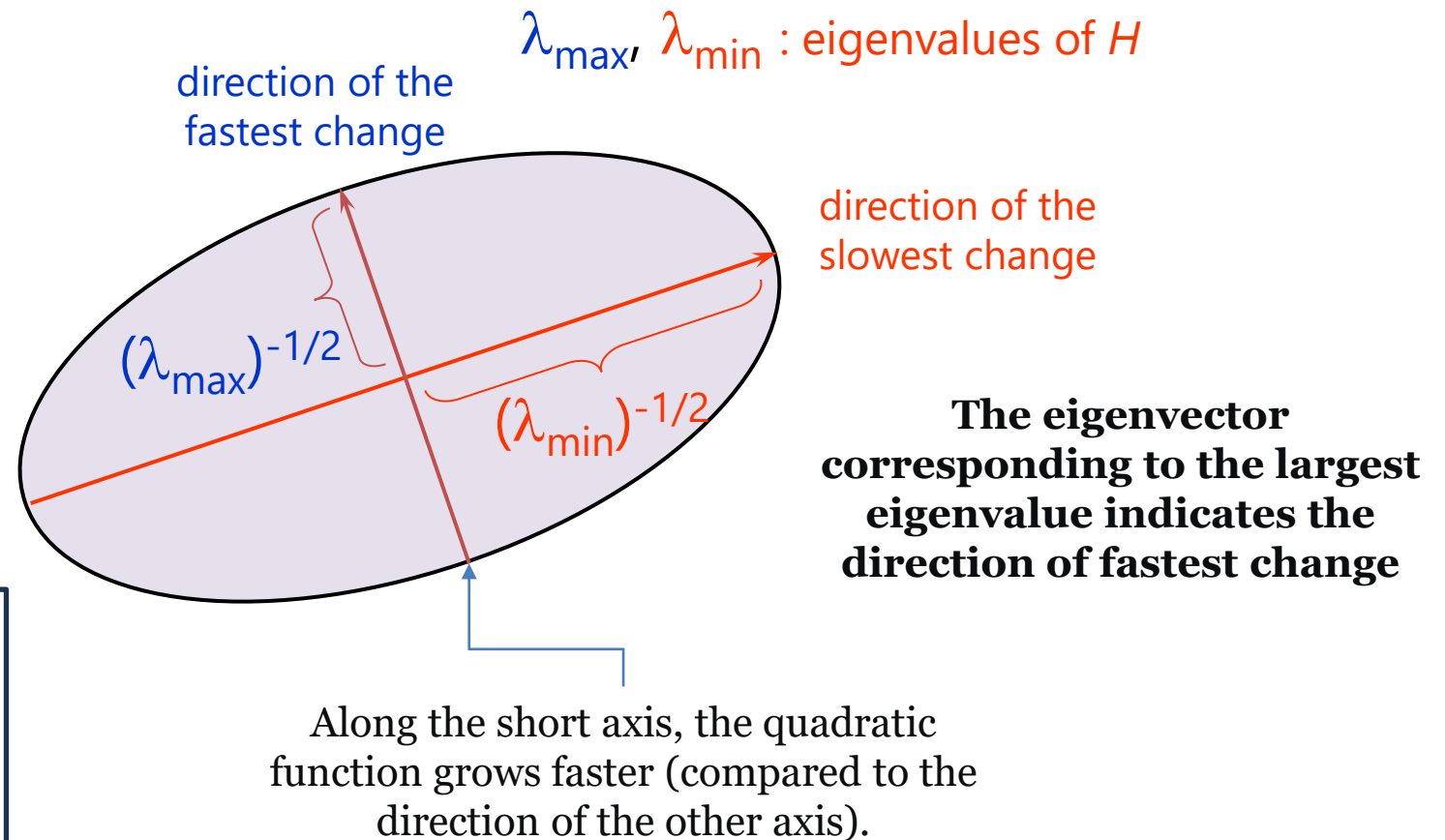
General case

We can visualize $E(u, v)$ as an ellipse with axis lengths determined by the *eigenvalues* of H and orientation determined by the *eigenvectors* of H

Ellipse equation:

$$\begin{bmatrix} u & v \end{bmatrix} H \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$

$$E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_H \begin{bmatrix} u \\ v \end{bmatrix}$$



Quick eigenvalue/eigenvector review

The **eigenvectors** of a matrix **A** are the vectors **x** that satisfy:

$$Ax = \lambda x$$

The scalar λ is the **eigenvalue** corresponding to **x**

- The eigenvalues are found by solving:

$$\det(A - \lambda I) = 0$$

- In our case, **A** = **H** is a 2x2 matrix, so we have

$$\det \begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} = 0$$

- The solution:

$$\lambda_{\pm} = \frac{1}{2} \left[(h_{11} + h_{22}) \pm \sqrt{4h_{12}h_{21} + (h_{11} - h_{22})^2} \right]$$

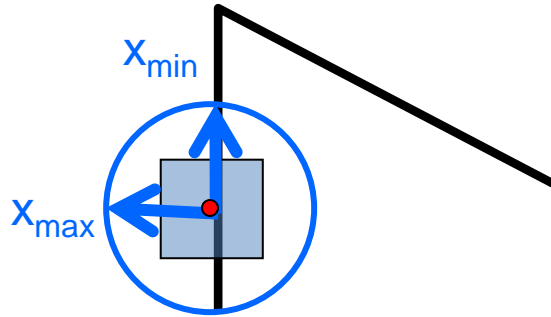
Does this sound familiar?

Once you know λ , you find **x** by solving

$$\begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0$$

Corner detection: the math

$$E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_H \begin{bmatrix} u \\ v \end{bmatrix}$$



The **eigenvectors** of a matrix **A** are the vectors **x** that satisfy: $Ax = \lambda x$



$$Hx_{\max} = \lambda_{\max}x_{\max}$$

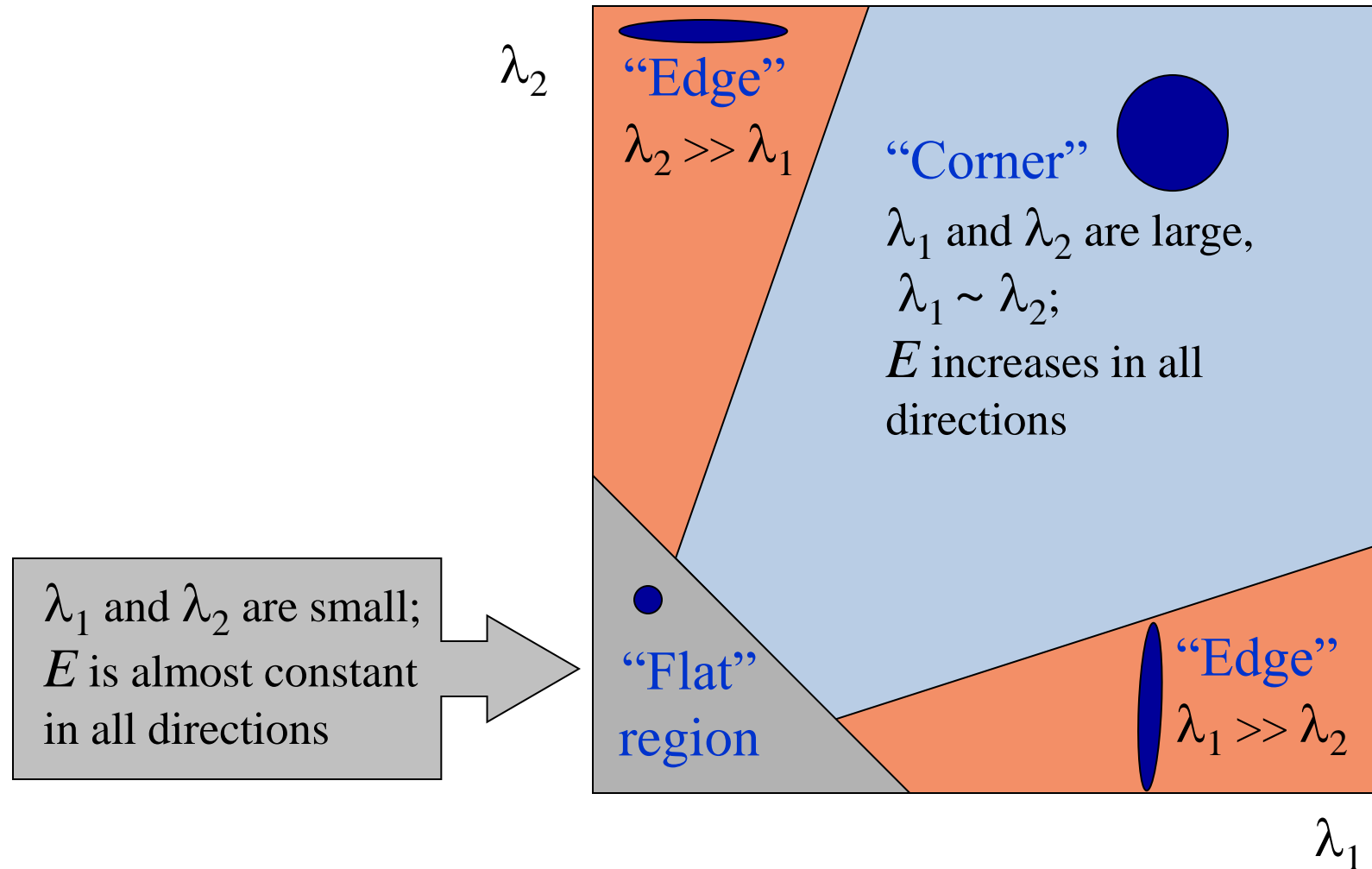
$$Hx_{\min} = \lambda_{\min}x_{\min}$$

Eigenvalues (λ_{\max} , λ_{\min}) and eigenvectors (x_{\max} , x_{\min}) of H

- **Define shift directions with the smallest and largest change in error**
- x_{\max} = direction of largest increase in E
- λ_{\max} = **amount of increase in direction x_{\max}**
- x_{\min} = direction of smallest increase in E
- λ_{\min} = **amount of increase in direction x_{\min}**

Interpreting the eigenvalues

Classification of image points using eigenvalues of H :



Harris Corner Detector or Harris Operator

- There are several definitions of the Harris operator (measure of *cornerness*):

$$R = \lambda_1 \lambda_2 - k \cdot (\lambda_1 + \lambda_2)^2 = \det(H) - k \cdot \text{trace}(H)^2$$

Original proposal by
Harris & Stephens (1988)

$$R = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} = \frac{\det(H)}{\text{trace}(H)}$$

Proposed by Brown,
Szeliski & Winder (2005)

$$R = \lambda_{\min} - \alpha \lambda_{\max}$$

Proposed by Triggs
(2004), with $\alpha = 0.05$

$$R = \lambda_{\min}$$

“Find maxima in the
smaller eigenvalue to
locate good features to
track” (Shi & Tomasi, 1994)

Note: The *trace* is the sum of the diagonals, i.e., $\text{trace}(H) = h_{11} + h_{22}$

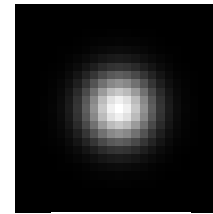
Weighting the derivatives

- In practice, using a simple window W doesn't work too well

$$H = \sum_{(x,y) \in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

- Instead, we'll *weight* each derivative value based on its distance from the center pixel

$$H = \sum_{(x,y) \in W} w_{x,y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$



$w_{x,y}$

Window function (Harris uses a Gaussian window)

Harris Detector [Harris88]

Second moment matrix:

$$H = \mu(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

Integration scale

Derivative scale

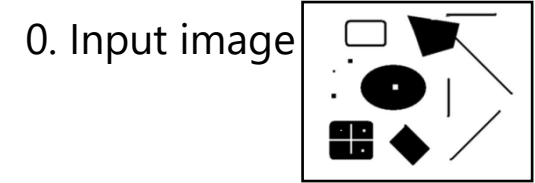
$$R = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} = \frac{\det(H)}{\text{trace}(H)}$$

2. Product/square of derivatives
(3 images corresponding to the 3 terms in matrix H)

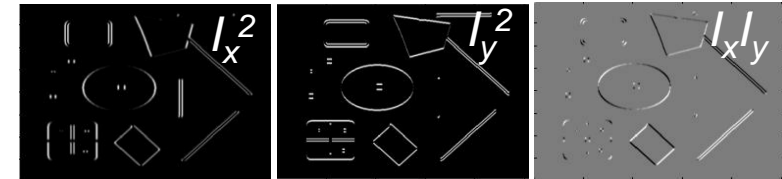
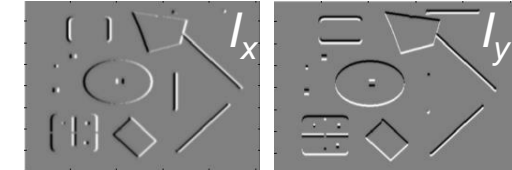
3. We convolve these 3 images with a Gaussian filter $g(\sigma_I)$

4. Compute scalar cornerness value using one of the R measures presented before (both eigenvalues are strong)

5. Non-maxima suppression



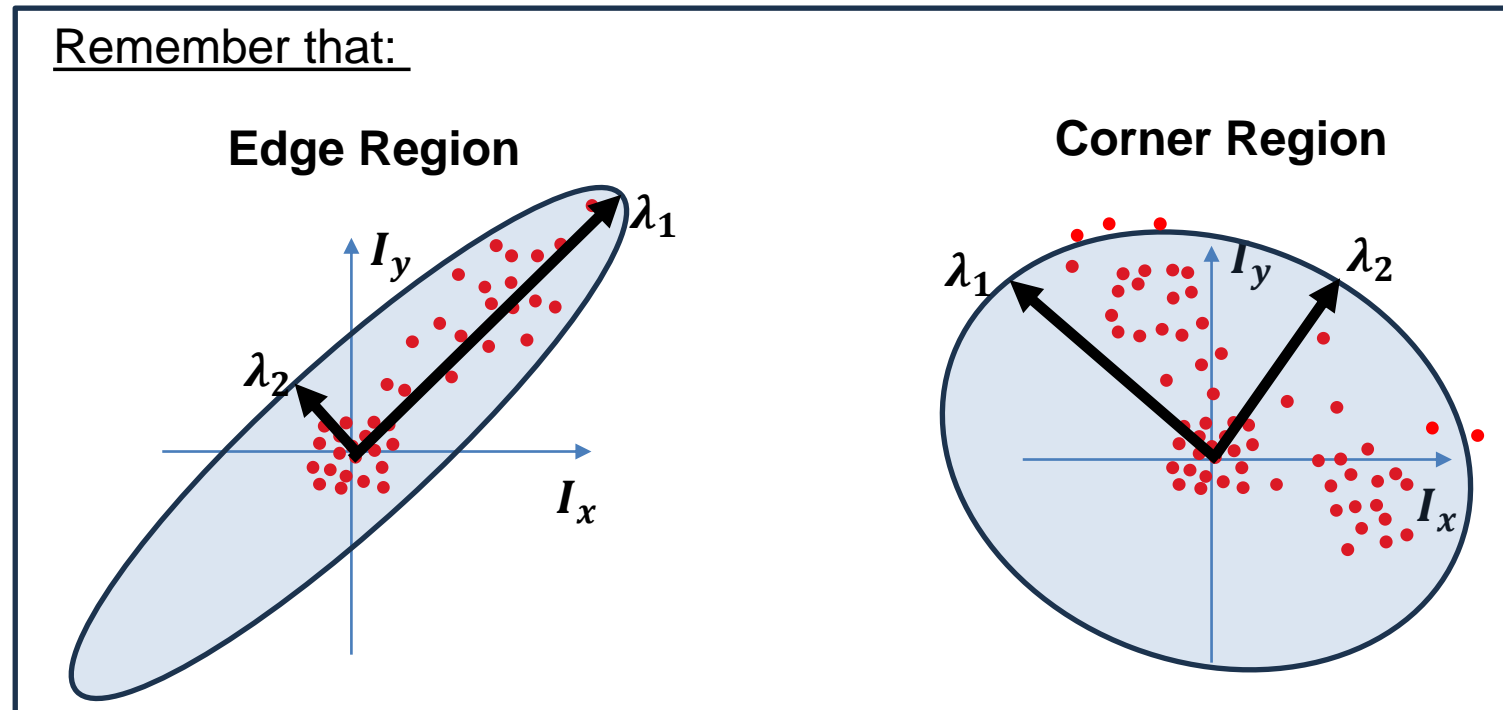
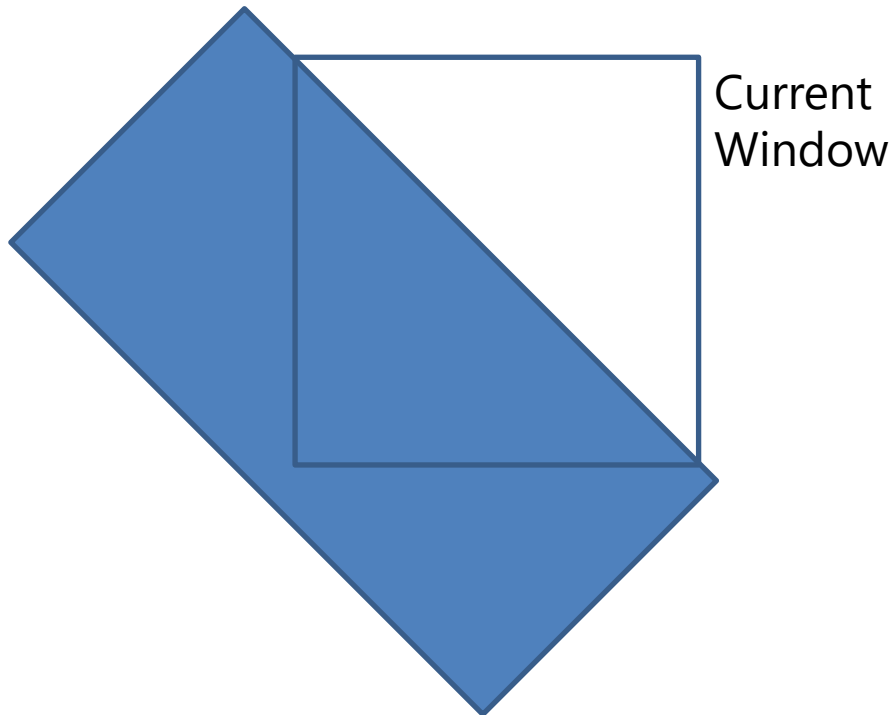
1. Image derivatives



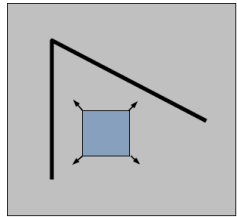
See slides 128-138 in
["Lecture 10: Edge Detection \(cont.\)"](#) by Leonid Sigal for a practical example of Harris Corner Detection.

Harris Corners – Why so complicated?

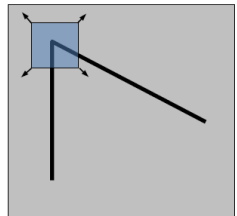
- Can't we just check for regions with lots of gradients in the x and y directions?
 - No! A diagonal line would satisfy that criterion!



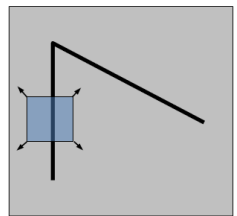
Harris detector



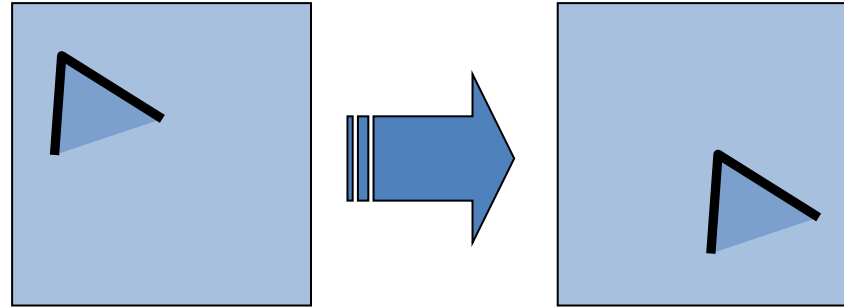
"flat" region:
no change in all
directions



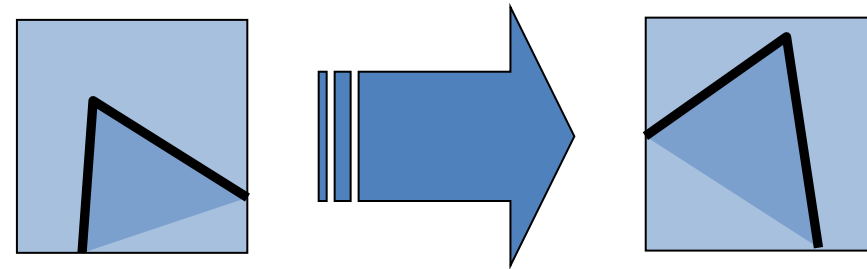
"corner":
significant change in
all directions



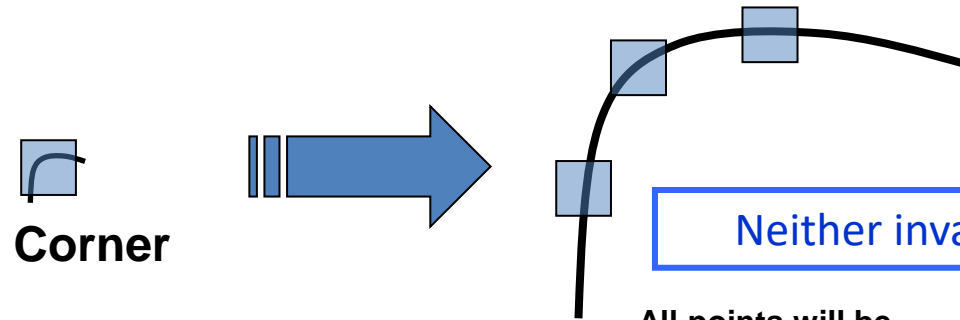
"edge":
no change along the
edge direction



Corner location is equivariant w.r.t. translation



Corner location is equivariant w.r.t. image rotation



Corner

Neither invariant nor equivariant to scaling

All points will be
classified as **edges**

And here SIFT
appears!!!!

By the way, this is
detection... what
about description???

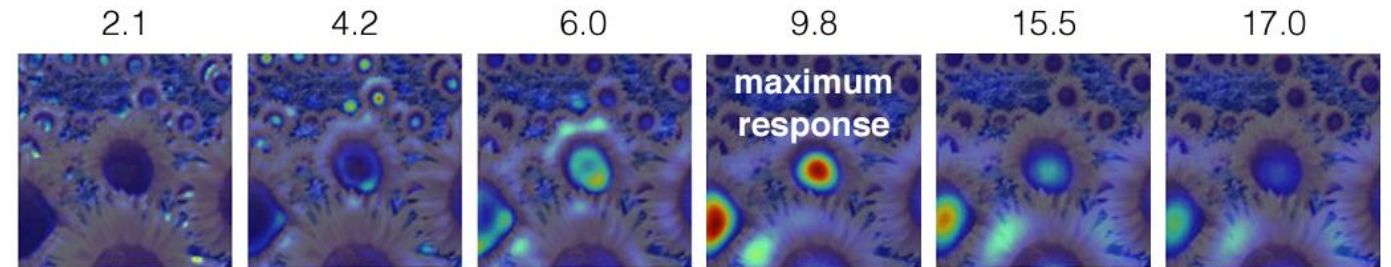
Scale-Invariant Feature Transform (SIFT)

Multi-scale 2D blob/corner detection

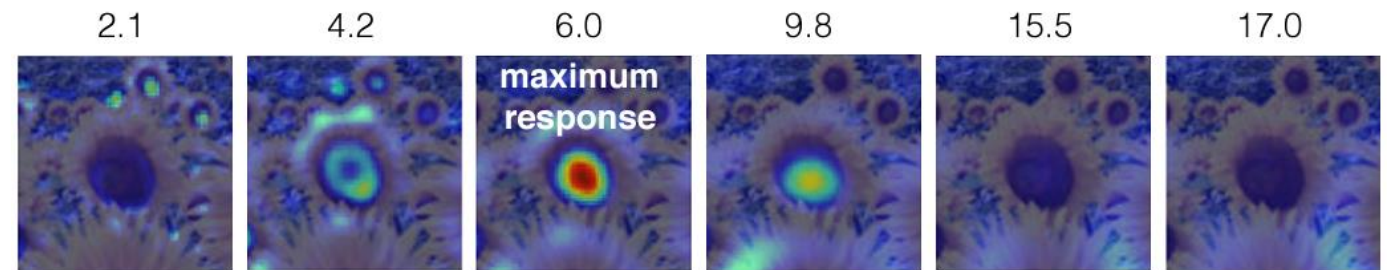
- Find local maxima in both position (space) and scale

For each level of the Gaussian pyramid
compute feature response (e.g. Harris, LoG)

For each level of the Gaussian pyramid
if cross-scale local maximum
save scale and location of feature (x, y, s)



Full size image



3/4 size image

SIFT (Scale Invariant Feature Transform)

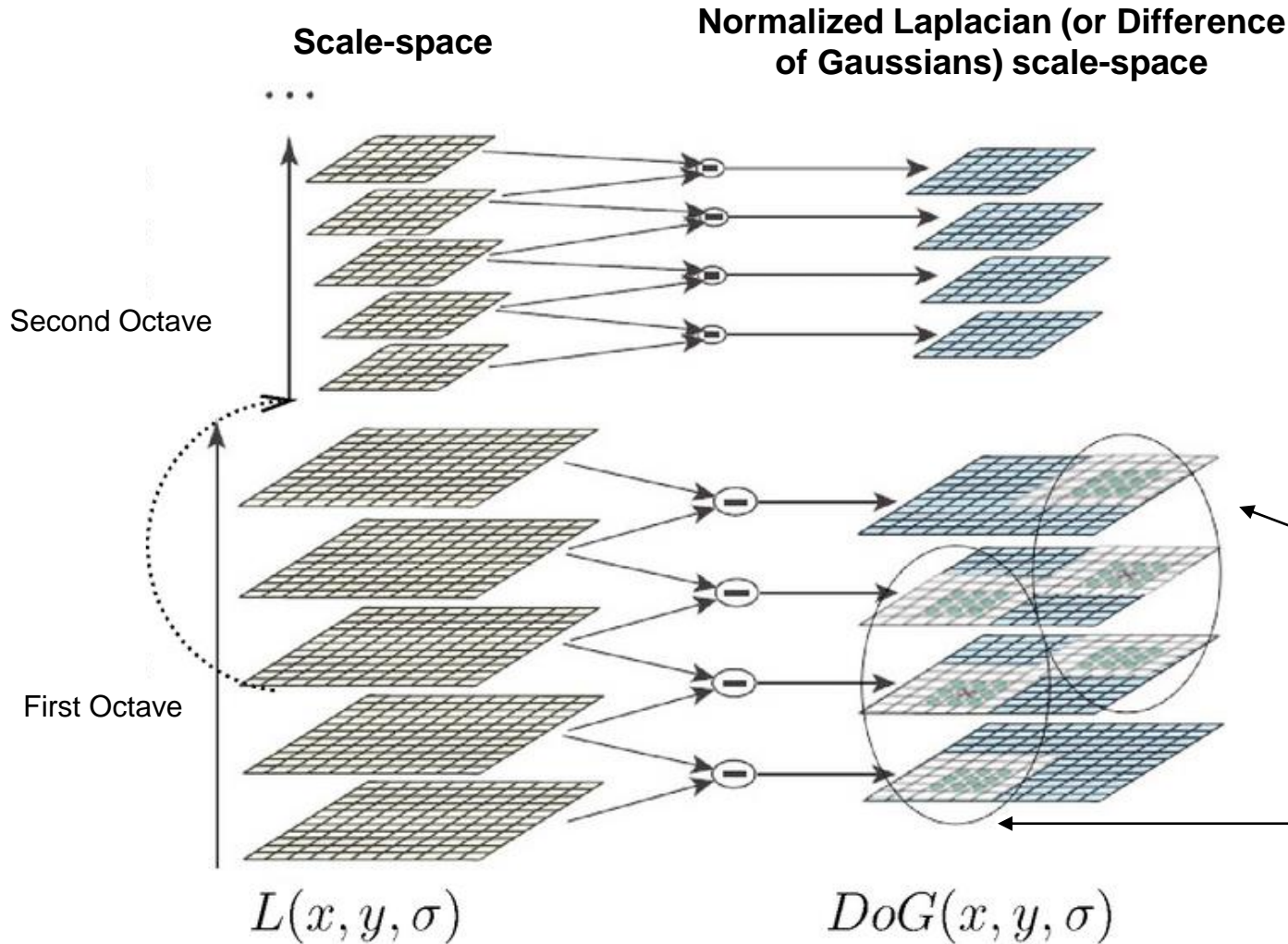
- Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60, 91-110. (>72.000 citations)
- Highly recommended reading:
 - "Anatomy of the SIFT Method" (Ives Rey-Otero, Mauricio Delbracio) Image Processing On Line, 2014 <http://dx.doi.org/10.5201/ipol.2014.82>
From Otero's PhD dissertation (École normale supérieure de Cachan-ENS Cachan): <https://theses.hal.science/tel-01226489/document>

SIFT detector

Overall, similar to multiscale blob detection.

Detect the 3D discrete maxima and minima (the number of **neighbors** is $26=3\times3\times3-1$).

This scanning process is **sensitive to noise** and produces **unstable detections** → This step is followed by an **interpolation**, that refines the localization of the extrema, and by filtering to **discard unreliable detections (edges)**.



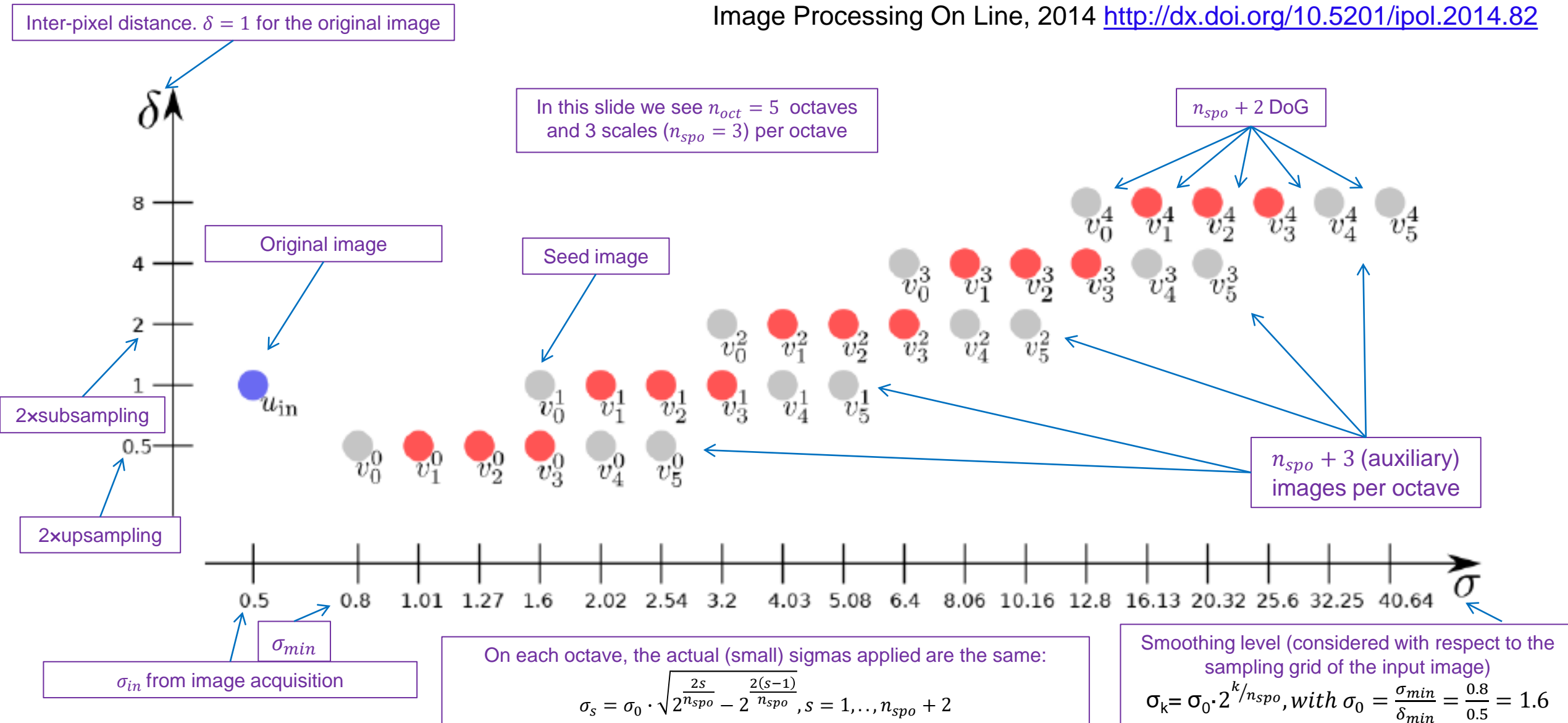
Octave: sets of images sharing a common sampling rate.

Each octave corresponds to a doubling of σ , just as an octave in music theory corresponds to doubling the frequency of a sound signal (González & Woods, 2018).

Many technical details related with the Gaussian scale-space and the keypoint definition are included in ["Anatomy of the SIFT Method"](#)

SIFT detector: Scale-space default configuration

"Anatomy of the SIFT Method" (Ives Rey-Otero, Mauricio Delbracio)
Image Processing On Line, 2014 <http://dx.doi.org/10.5201/ipol.2014.82>



SIFT detector: The DoG scale-space

"Anatomy of the SIFT Method" (Ives Rey-Otero, Mauricio Delbracio)
Image Processing On Line, 2014 <http://dx.doi.org/10.5201/ipol.2014.82>

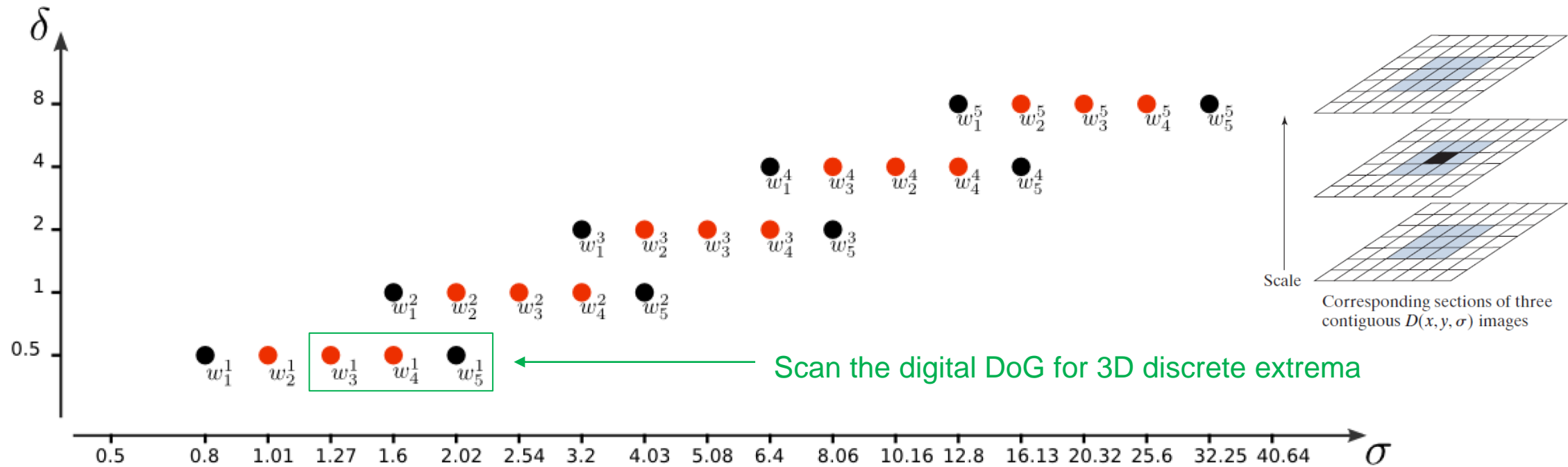
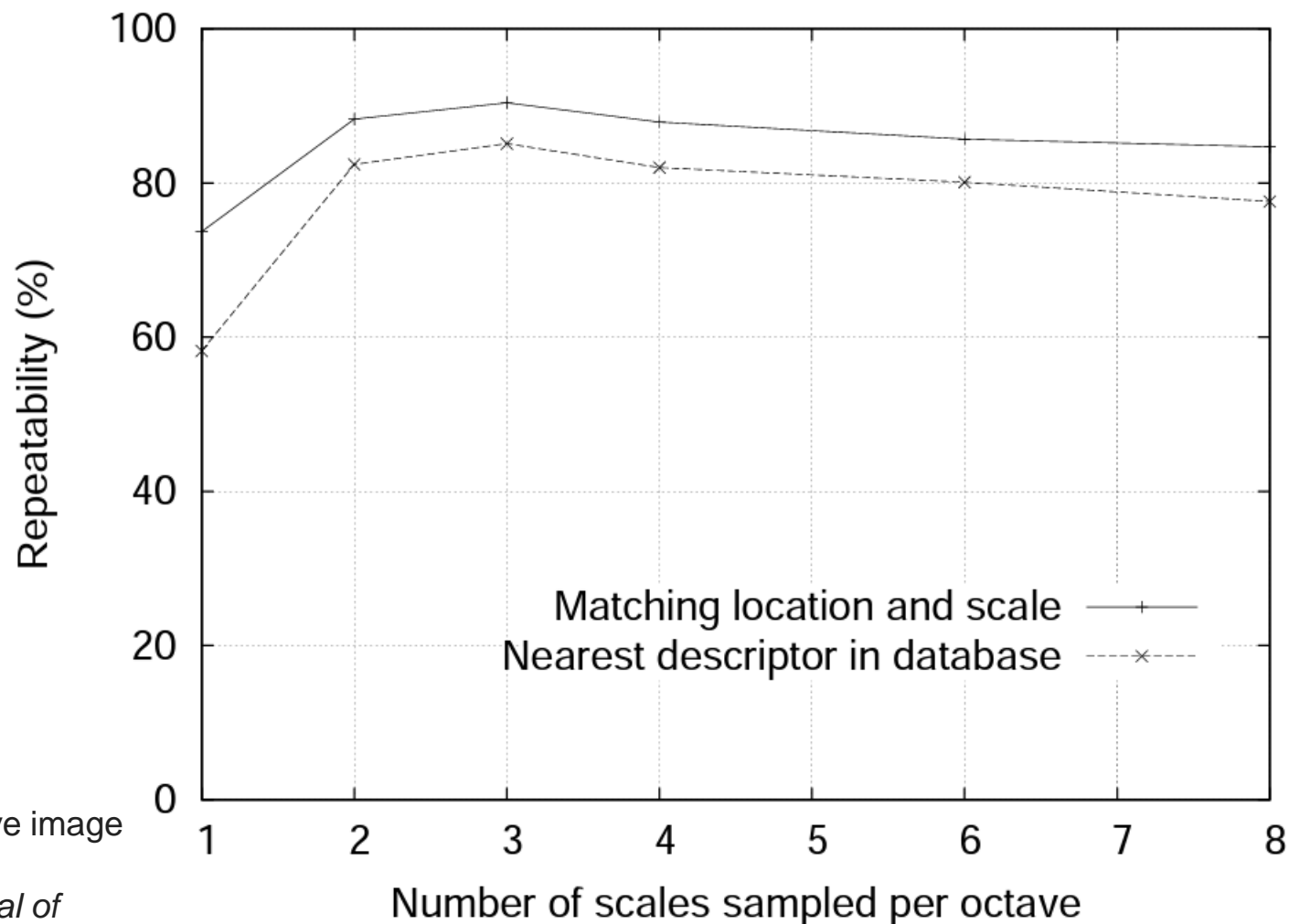


Figure 5: The *DoG* scale-space. The difference of Gaussians acts as an approximation of the normalized Laplacian $\sigma^2 \Delta$. The difference $\mathbf{w}_s^o = \mathbf{v}_{s+1}^o - \mathbf{v}_s^o$ is relative to the blur level σ_s^o . Each octave contains n_{spo} images plus two auxiliary images (in black).

SIFT detector: Scale-space default configuration



SIFT is **strongly heuristic**: it uses a significant number of **experimentally determined parameters**.

SIFT orientation assignment

- Take a square window around detected feature (in the corresponding scale)
- Compute edge orientation (gradient direction for each pixel)
- Create a weighted direction histogram (36 bins: 10° per bin). Weights are:
 - Gradient magnitudes

Arrows illustrate gradient orientation (**arrow direction**) and gradient magnitude (**arrow length**)

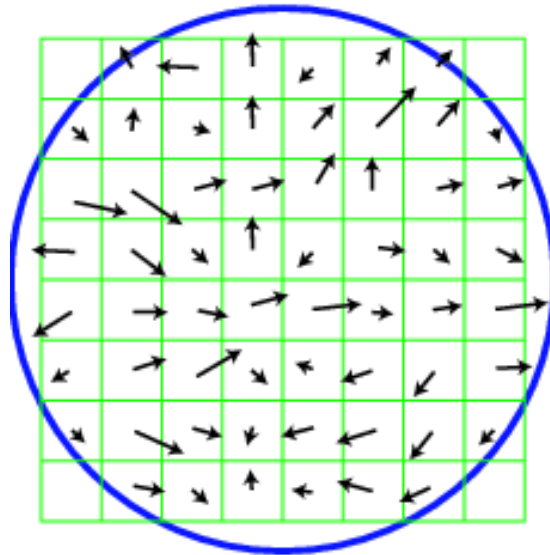
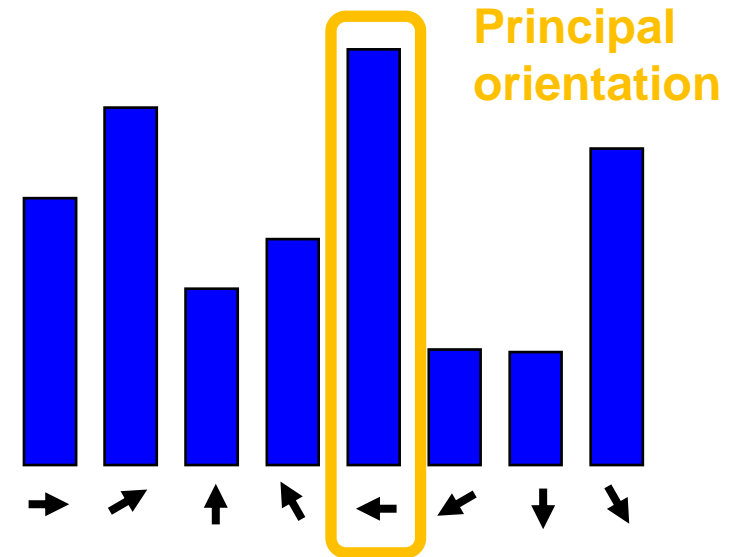


Image gradients



SIFT orientation assignment

- Take a square window around detected feature (in the corresponding scale)
- Compute edge orientation (gradient direction for each pixel)
- Create a weighted direction histogram (36 bins: 10° per bin). Weights are:
 - Gradient magnitudes
 - Spatial Gaussian filter with $\sigma = 1.5 \cdot \langle \text{keypoint_scale} \rangle$
(central pixels more important)

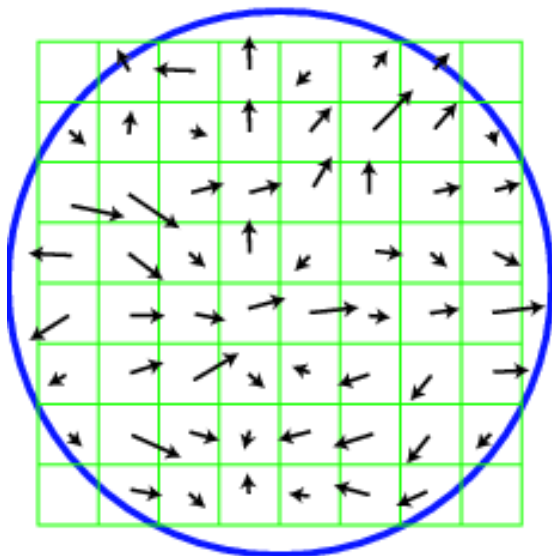
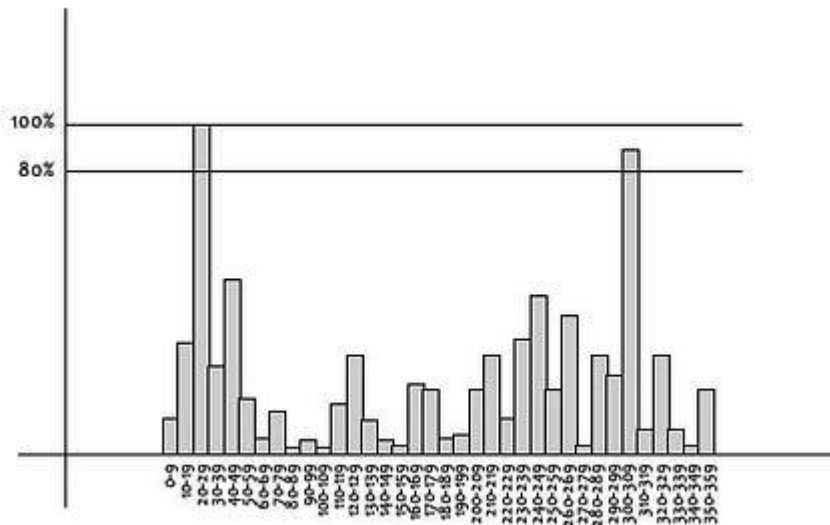


Image gradients



Peaks in the orientation histogram correspond to **dominant/reference directions** of local gradients.

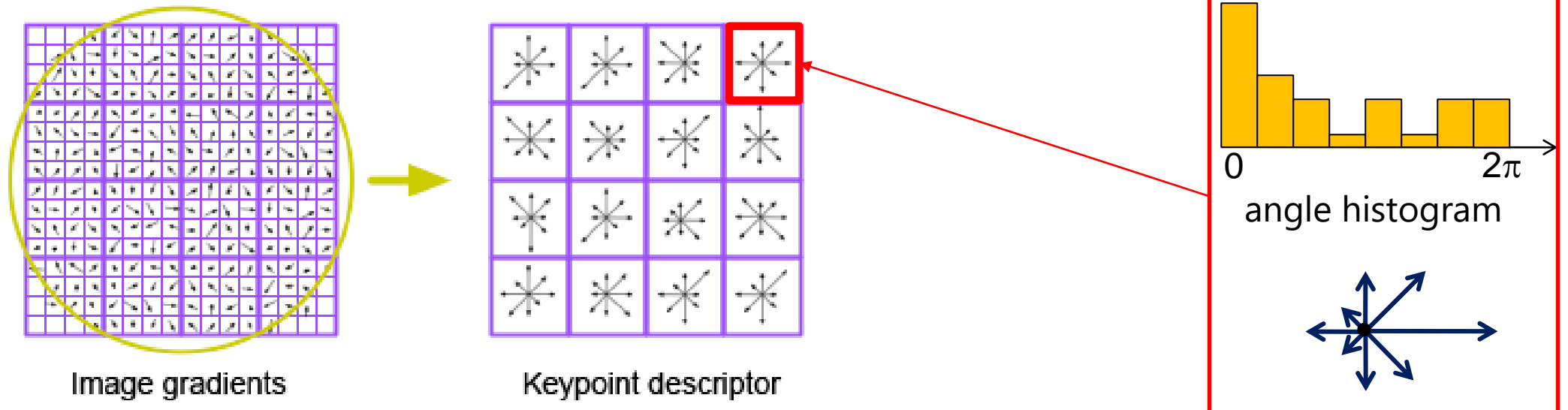
The highest peak is detected, and also any other local peak within 80% of the highest peak is used to create a keypoint with that orientation.

Each keypoint: **(x, y, scale, orientation)**

SIFT descriptor

For each (x, y, scale, orientation), create descriptor:

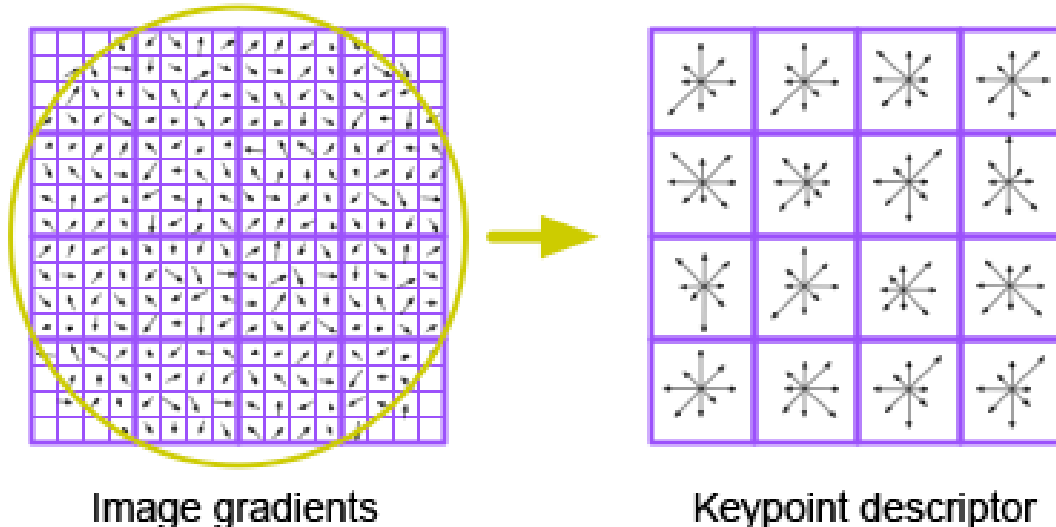
- Using a normalized patch after undoing the effect of rotation and scaling (orientation and scale invariance)
- Divide a 16x16 window around the keypoint location into a 4x4 grid of cells
- Compute a weighted orientation histogram for each cell (8 bins)
- Final descriptor: 16 cells * 8 orientations = **128 dimensional descriptor**
- Descriptor is normalized to unit length ($\vec{v}/\|\vec{v}\|$), then clipped (if >0.2 is saturated to 0.2), and finally renormalized to unit length to reduce the effects of illumination change.



SIFT descriptor

For each (x, y, scale, orientation), create descriptor:

- Using a normalized patch after undoing the effect of rotation and scaling (orientation and scale invariance)
- Divide a 16x16 window around the keypoint location into a 4x4 grid of cells
- Compute a weighted orientation histogram for each cell (8 bins)
- Final descriptor: 16 cells * 8 orientations = **128 dimensional descriptor**
- Descriptor is normalized to unit length ($\vec{v}/\|\vec{v}\|$), then clipped (if >0.2 is saturated to 0.2), and finally renormalized to unit length to reduce the effects of illumination change.



Main contribution of SIFT: the descriptor!

Before, you usually computed gray-level statistics. Since SIFT works with gradient orientations, it can operate adequately with strong changes in illumination.

Properties of SIFT

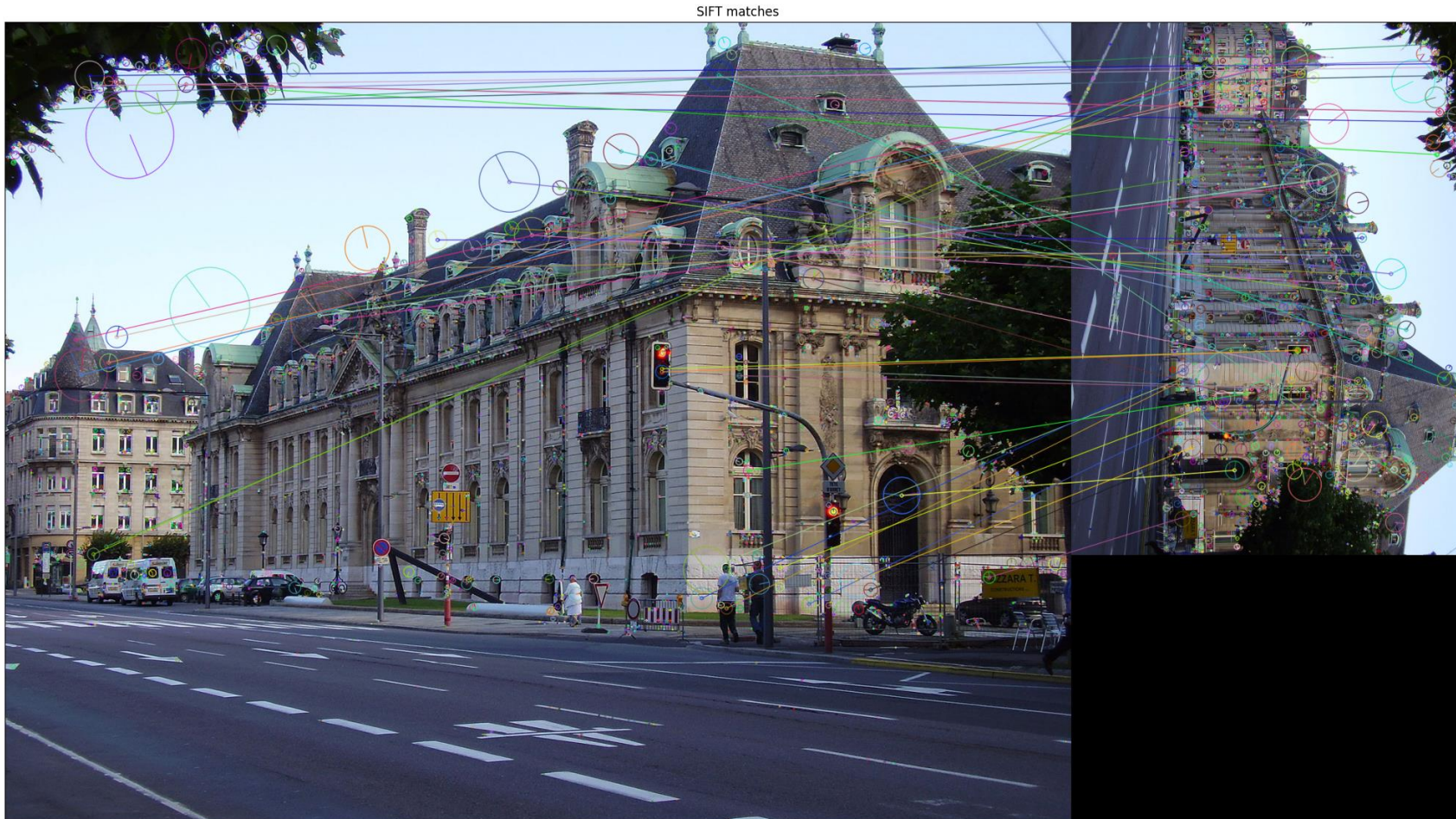
Extraordinarily robust matching technique

- Can handle relatively large changes in viewpoint (up to about 30-60 degrees, depending on the authors)
- Can handle significant changes in illumination (sometimes even day vs. night (below))
- Pretty fast (hard to make real-time, but can run in $<1s$ for moderate image sizes)
- Lots of code available



Properties of SIFT

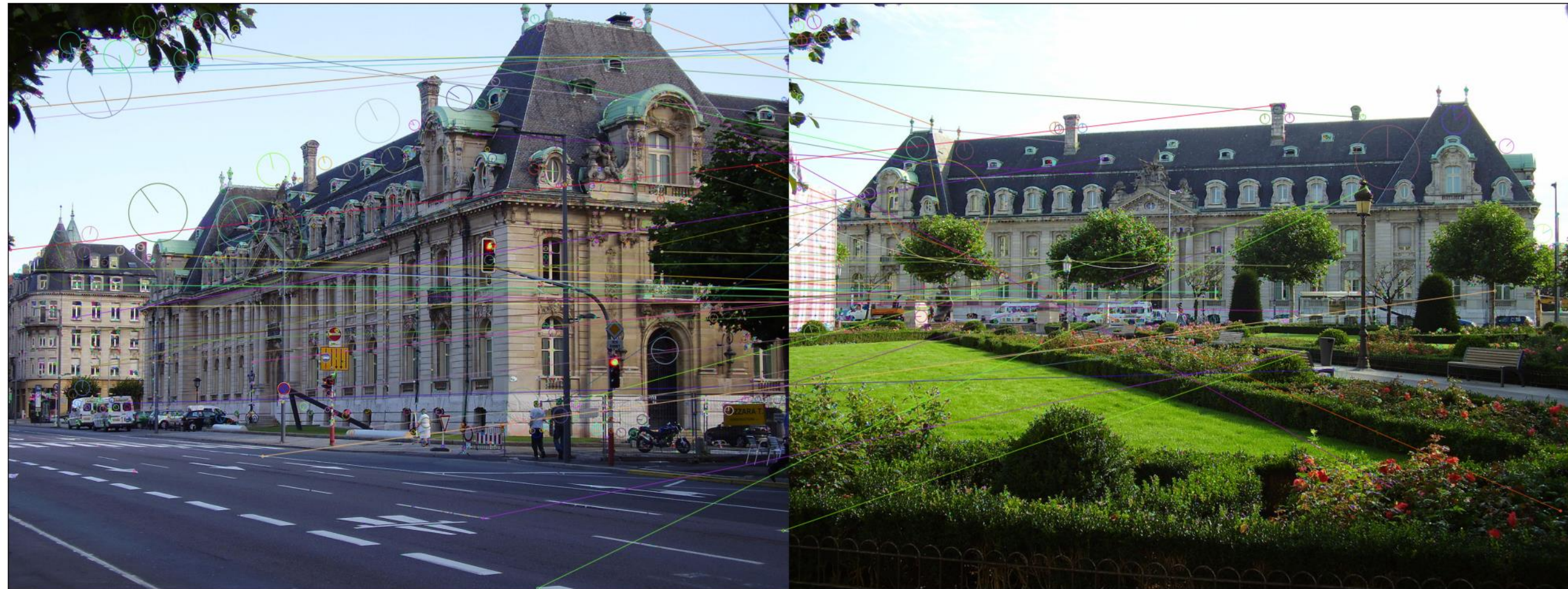
Invariant to translation, rotation and scale, and robust to clutter and changes in brightness. In the example below, we applied SIFT to two images (one is the scaled and 90° rotated version of the other one). We just display 50 matches.



Properties of SIFT

If viewpoint differs a lot we could have problems...

SIFT matches

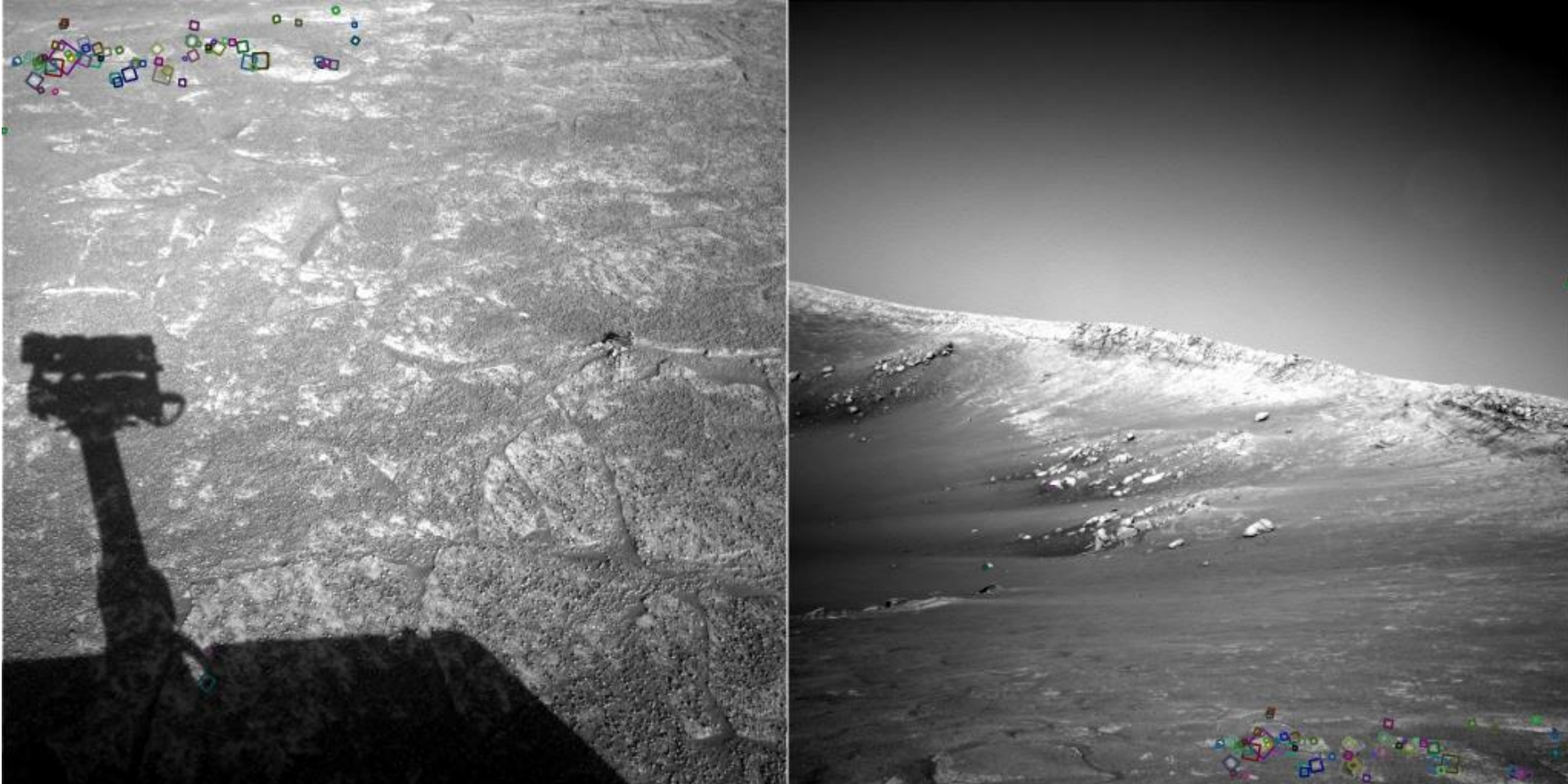


Another SIFT example



The size of the blob is proportional to the scale where it was found

Very stable in difficult scenes



NASA Mars Rover images
with SIFT feature matches

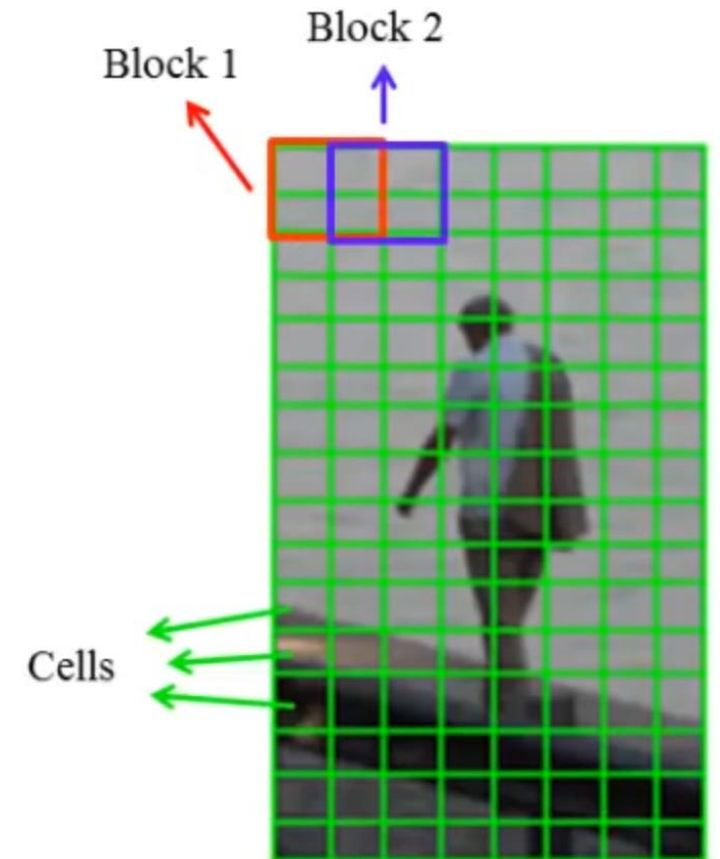
Histogram of Oriented Gradients (HOG)

HOG (Histogram of Oriented Gradients)

- Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *IEEE computer society conference on computer vision and pattern recognition (CVPR'05)* (Vol. 1, pp. 886-893). (>44.000 citations)
- Similar to SIFT, but it is computed on a dense grid of uniformly spaced cells → this is a global descriptor!
- The original work was focused on pedestrian detection.

HOG Feature Extraction

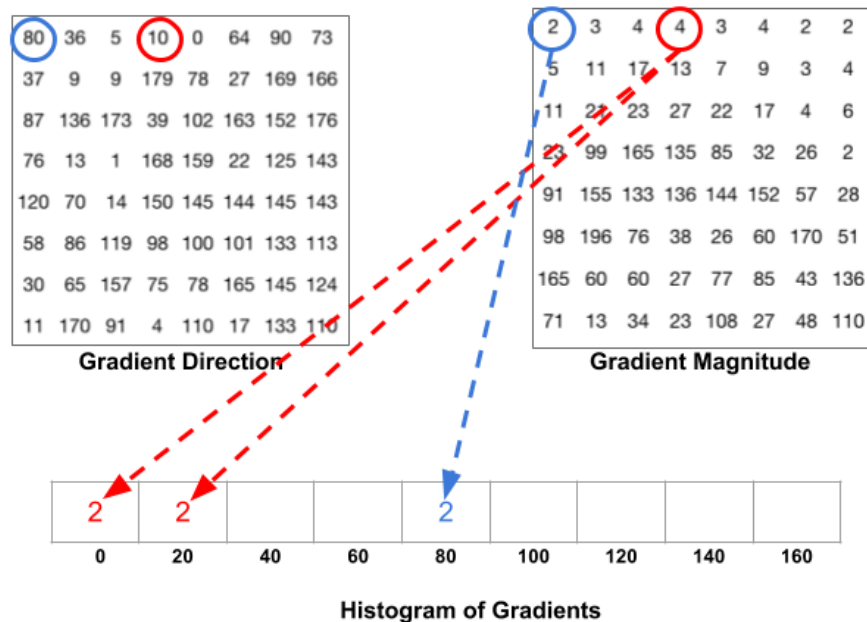
- 1) Compute gradients of the input image (64x128 in the original paper)
- 2) Create 16x16 pixels overlapping blocks
 - $7 \times 15 = 105$ blocks in total
 - Each block should consist of 2×2 cells with size 8×8 (each cell)



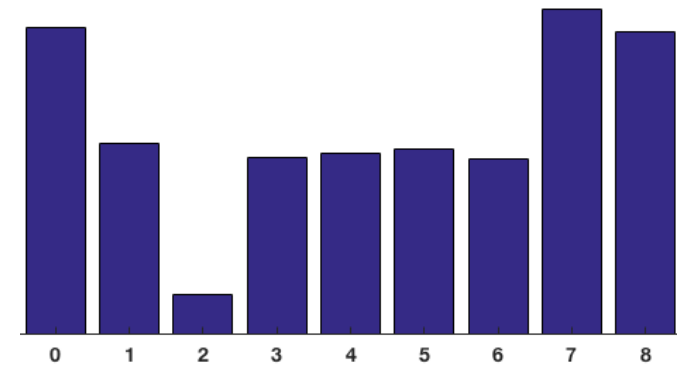
HOG Feature Extraction

3) Quantize the gradient orientation of each cell into a 9 bins/orientations histogram

- The vote is the gradient magnitude.
- Interpolate votes linearly between neighboring bin centers.
- The vote can also be weighted with Gaussian to downweight the pixels near the edges of the block.



Since 10 degrees is halfway between 0° and 20° , the vote by the pixel splits evenly into the two bins.

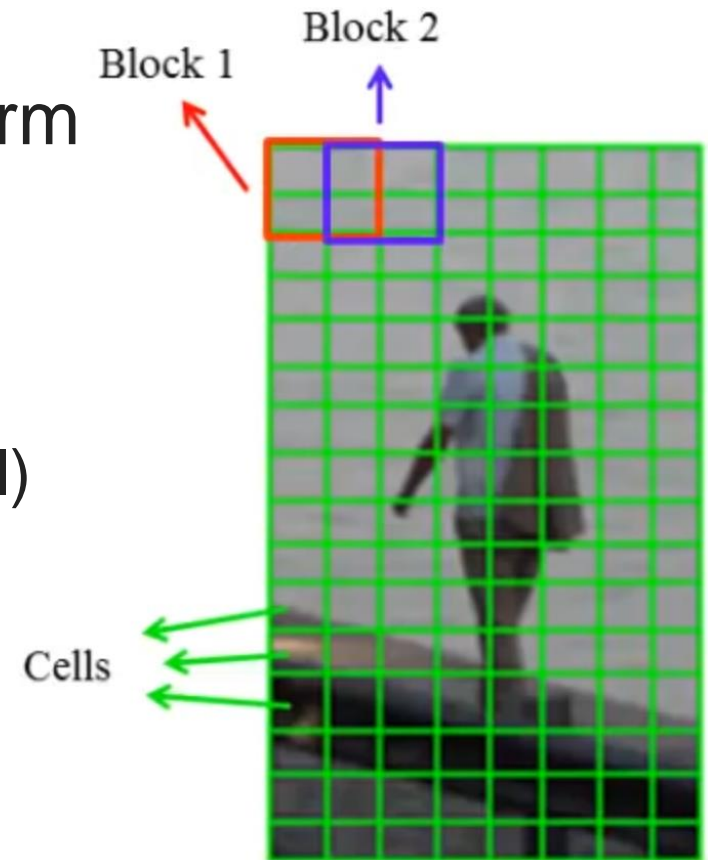


Example taken from

<https://learnopencv.com/histogram-of-oriented-gradients/>

HOG Feature Extraction

- 4) Normalize histograms on each block (9 bins x 4 cells/block \rightarrow 36-d vector) to reduce the effects of illumination/lighting variation
 - We divide each vector element by the L2 norm of the vector.
- 5) Concatenate histograms (we have one per cell)
 - 105 blocks x 4 cells x 9 orientations
= 3.780 dimensional feature vector



Other feature detectors/descriptors

Some Feature Detectors/Descriptors

- SURF: Speeded up robust features
 - Bay et al. (ECCV'06) (>23.000 citations)
 - 6 times faster than SIFT
 - Equivalent quality for object identification
- ORB
 - Rublee et al. (ICCV'11) (>12.000 citations)
 - FAST detector + BRIEF descriptor
 - Fast alternative to SIFT
- FAST
 - Rosten & Drummond (ECCV'06) (>6.500 citations)
 - Computationally efficient corner detector
- BRISK
 - Leutenegger et al. (ICCV'11) (>4.500 citations)
 - Binary descriptor
 - An order of magnitude faster than SURF in some cases
- FREAK: Fast Retina Keypoint
 - Alahi et al. (CVPR'12) (>2.500 citations)
 - Perceptually motivated
 - Can run in real-time
- KAZE features
 - Alcantarilla et al. (ECCV'12) (>1.500 citations)
 - Nonlinear scale space by means of nonlinear diffusion filtering
- AKAZE features
 - Alcantarilla et al. (BMVC'13) (>1.350 citations)
 - Fast KAZE features
- LIFT: Learned Invariant Feature Transform
 - Yi et al. (ECCV'16) (>1.250 citations)
 - Learned via deep learning
- BRIEF
 - Calonder et al. (TPAMI'11) (>1.000 citations)
 - Very fast binary descriptor.
 - Comparable performance to SURF and SIFT

Some Feature Detectors/Descriptors

Feature	Detection	Description	OpenCV	Published
Harris	Yes	No	Yes	1988
KLT	Yes	No	Yes	1994
LBP	No	Yes	Yes	1994
SIFT	Yes	Yes	Yes	IJCV 2004
FAST	Yes	No	Yes	ECCV 2006
SURF	Yes	Yes	Yes	CVIU 2008
BRIEF	No	Yes	~	ECCV 2010
ORB	Yes	Yes	Yes	ICCV 2011
BRISK	Yes	Yes	Yes	ICCV 2011
FREAK	Yes	Yes	Yes	CVPR 2012

Table by Ajmal Mian & Mehdi Ravanbakhsh, "[Feature Detection and Extraction](#)"

Handcrafted Feature Detection and Extraction

Pablo Mesejo

pmesejo@go.ugr.es

Universidad de Granada

Departamento de Ciencias de la Computación e Inteligencia Artificial



UNIVERSIDAD
DE GRANADA



DaSCI

Instituto Andaluz de Investigación en
Data Science and Computational Intelligence