Indian Institute of Information Technology, Allahabad

CVBL IIIT Allahabad

Feature Extraction

Local Feature Extraction and Description

By

Dr. Satish Kumar Singh & Dr. Shiv Ram Dubey
Computer Vision and Biometrics Lab
Department of Information Technology
Indian Institute of Information Technology, Allahabad

TEAM

Computer Vision and Biometrics Lab (CVBL)

Department of Information Technology

Indian Institute of Information Technology Allahabad

Course Instructors

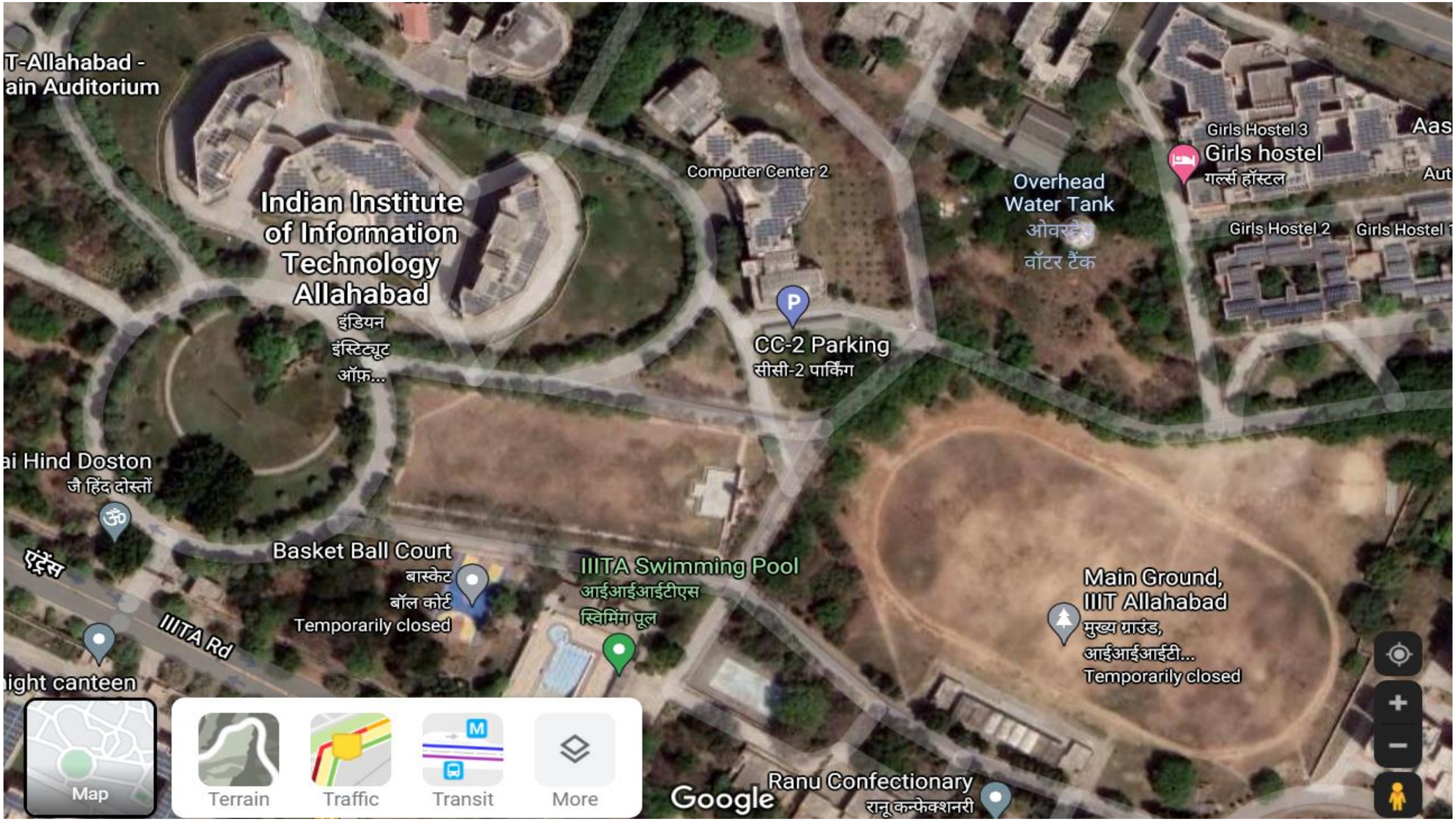
Dr. Satish Kumar Singh, Associate Professor, IIIT Allahabad (Email: sk.singh@iiita.ac.in)

Dr. Shiv Ram Dubey, Assistant Professor, IIIT Allahabad (Email: srdubey@iiita.ac.in)

DISCLAIMER

The content (text, image, and graphics) used in this slide are adopted from many sources for Academic purposes. Broadly, the sources have been given due credit appropriately. However, there is a chance of missing out some original primary sources. The authors of this material do not claim any copyright of such material.

INTEREST POINTS

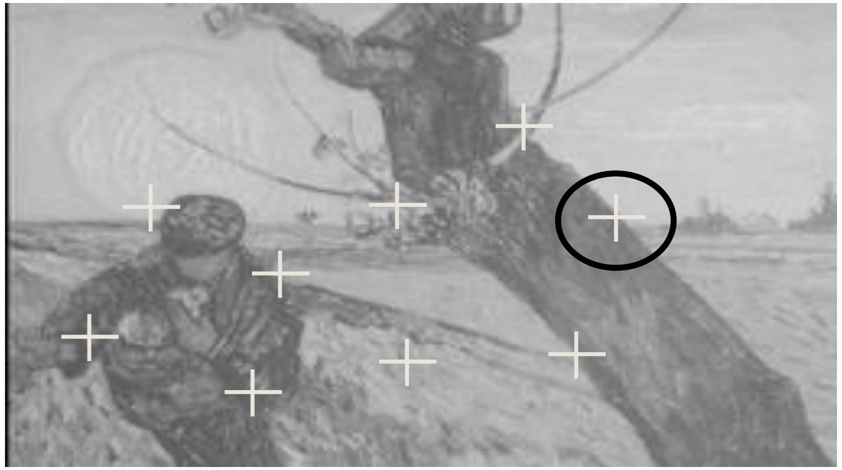


WHAT ARE LOCAL FEATURES?

- A pattern or distinct structure found in an image,
  - A point,
  - An edge,
  - A small patch
- The pattern or distinct structure differs from its immediate surroundings by
  - Texture,
  - Color,
  - Intensity
- Examples of local features
  - Corners,
  - Edge pixels,
  - Blobs

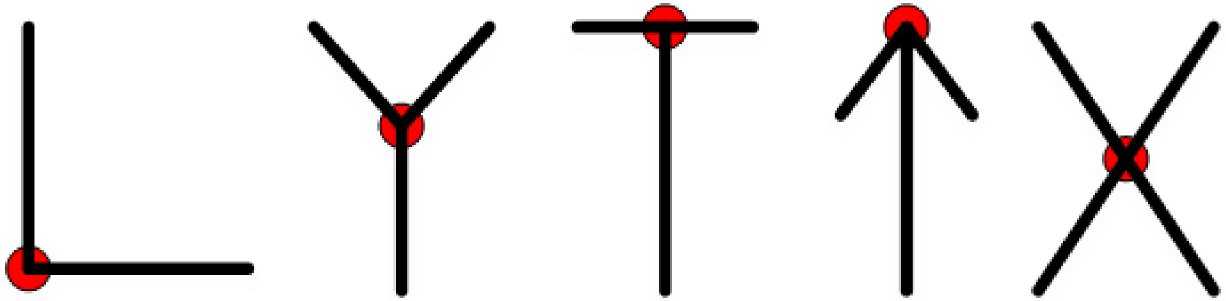
INTEREST POINTS

- A point in an image which has a well-defined position and can be robustly detected.
- Typically associated with a significant change of one or more image properties simultaneously (e.g., intensity, color, texture).



INTEREST POINTS AND CORNERS

- A corner can be defined as the intersection of two or more edges (special case of interest points).
- In general, interest points could be:
  - Isolated points of local intensity maximum or minimum.
  - Line endings.
  - Points on a curve where the curvature is locally maximized.



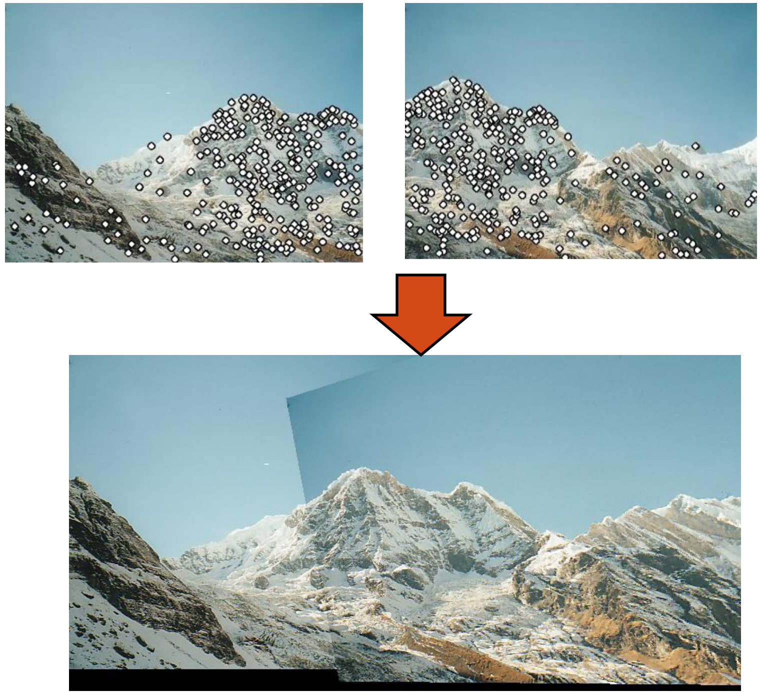
WHY ARE INTEREST POINTS USEFUL?

- For establishing corresponding points between images.

stereo matching

left camera right camera

panorama stitching

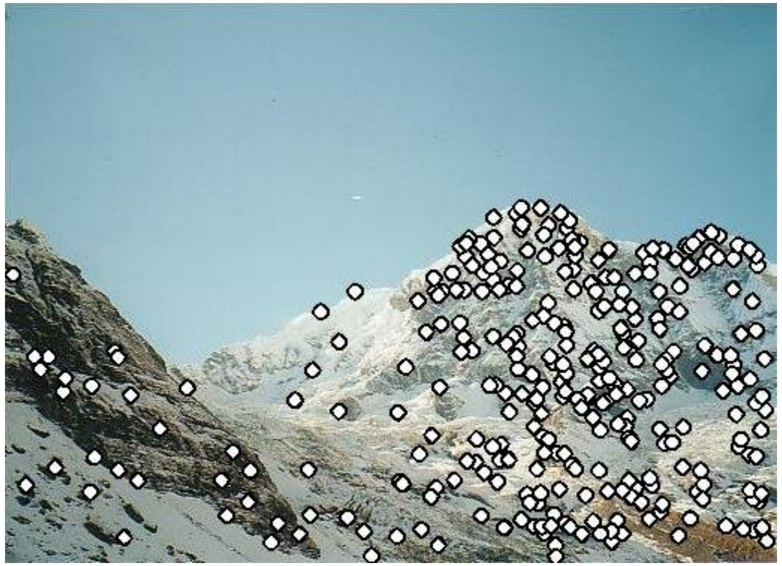


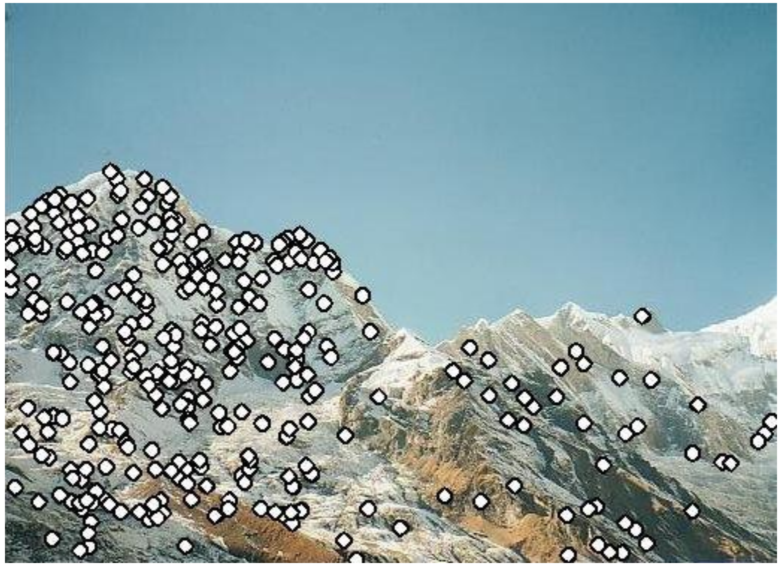
WHY EXTRACT FEATURES?

- Motivation: panorama stitching
  - We have two images – how do we combine them?

WHY EXTRACT FEATURES?

- Motivation: panorama stitching
  - We have two images – how do we combine them?

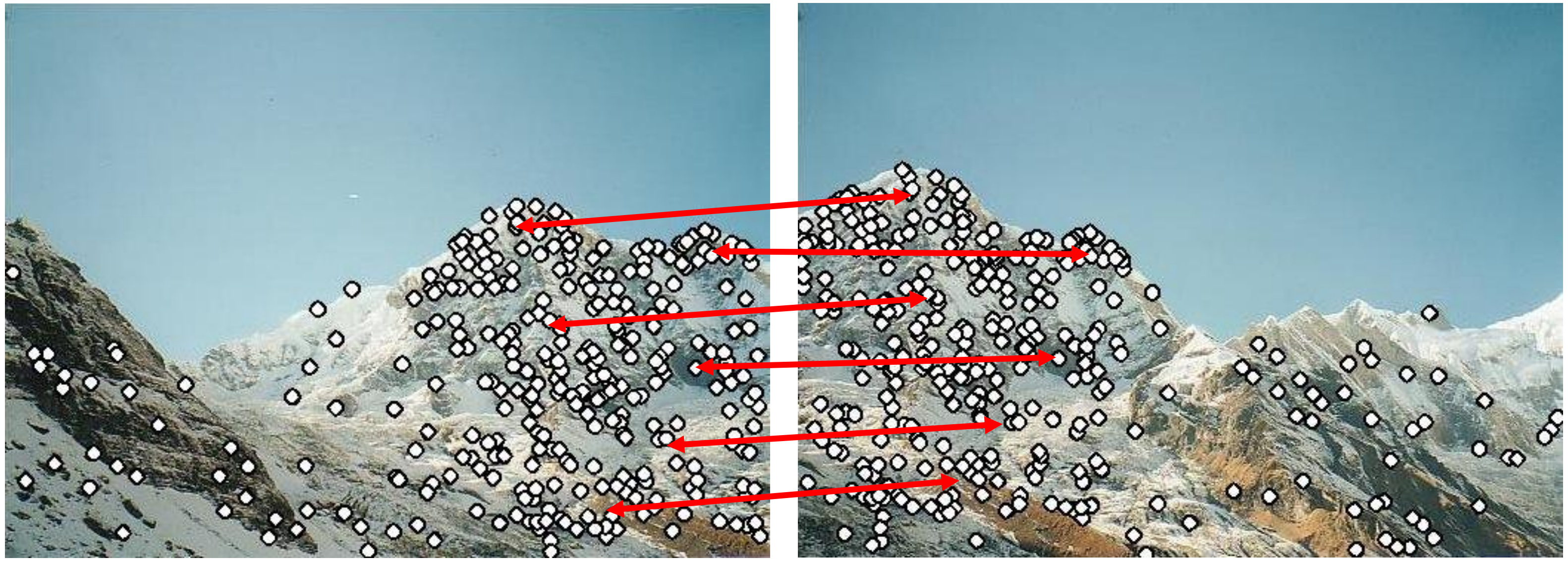




Step 1: extract features

WHY EXTRACT FEATURES?

- Motivation: panorama stitching
  - We have two images – how do we combine them?



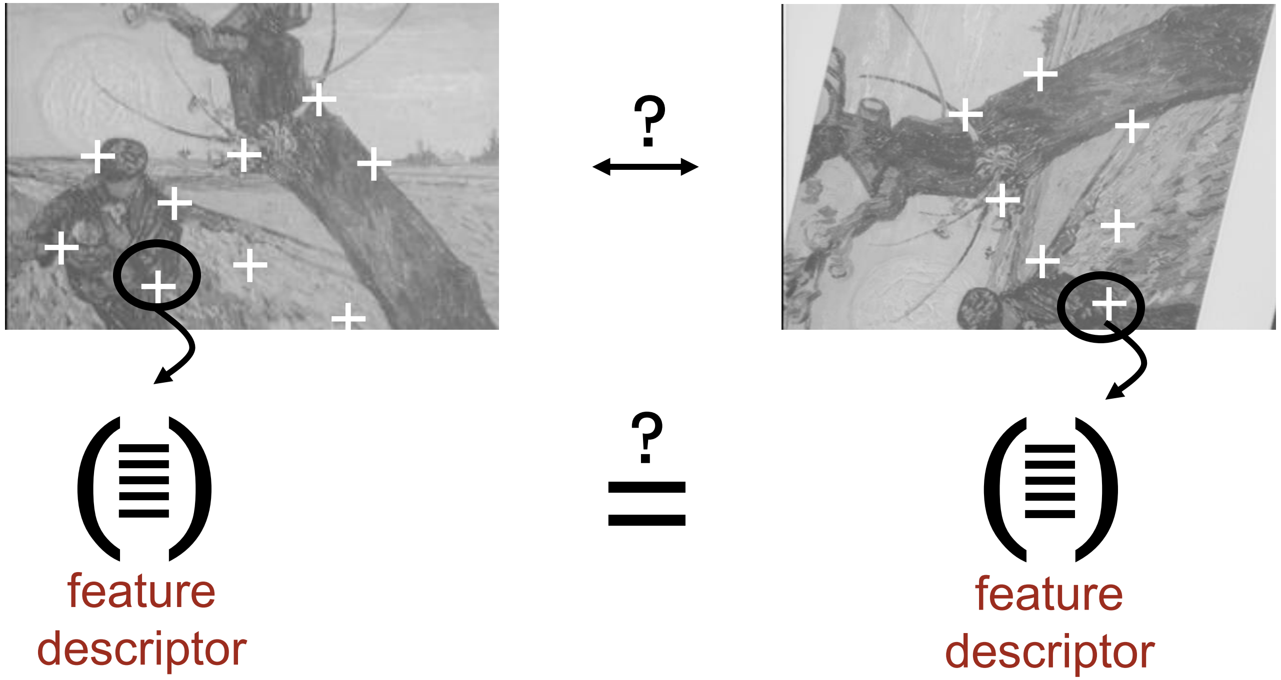
Step 1: extract features
Step 2: match features

WHY EXTRACT FEATURES?

- Motivation: panorama stitching
  - We have two images – how do we combine them?

Step 1: extract features
Step 2: match features
Step 3: align images

HOW COULD WE FIND CORRESPONDING POINTS?



- Need to define local patches surrounding the interest points and extract feature descriptors from every patch.
- Match feature descriptors to find corresponding points.

PROPERTIES OF GOOD FEATURES

- **Local:** features are local, so robust to occlusion and clutter (no prior segmentation!).
- **Accurate:** precise localization.

- **Invariant** (or **covariant**)
- **Robust:** noise, blur, compression, etc.

  do not have a big impact on the feature.

} **Repeatable**

- **Distinctive:** individual features can be matched to a large database of objects.
- **Efficient:** close to real-time performance.

INVARIANCE / COVARIANCE

- A function f is **invariant** under some transformation T if its value does not change when the transformation is applied to its argument:

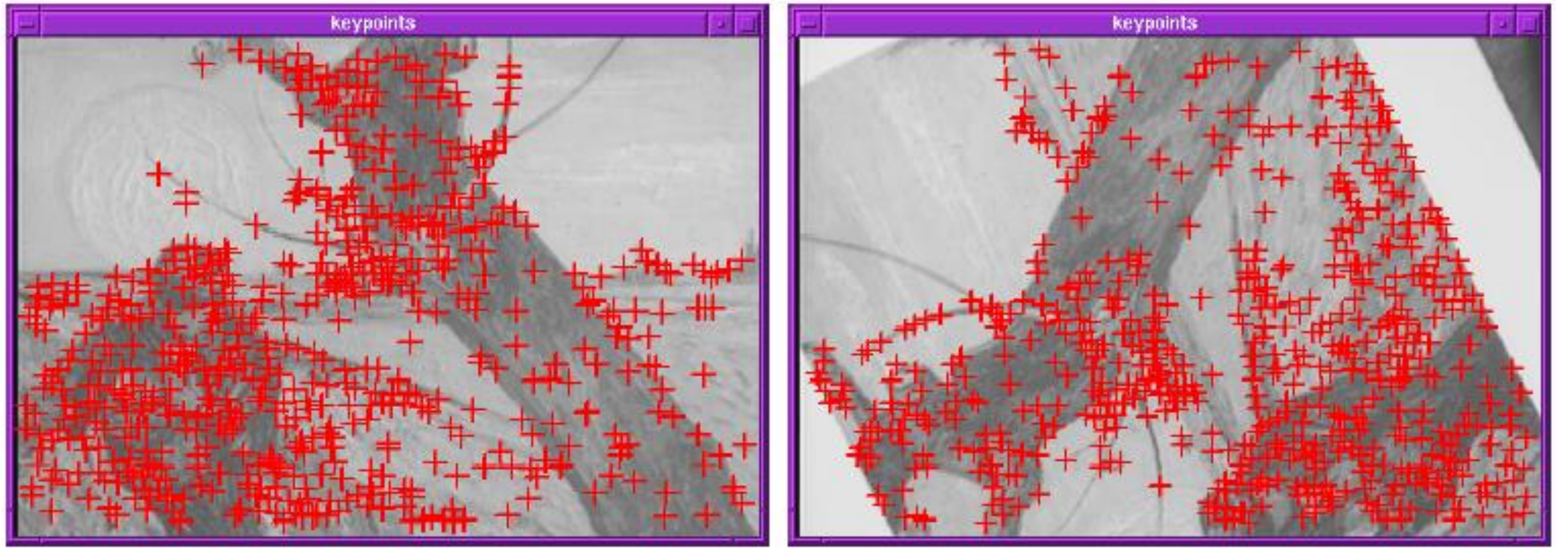
$$\text{if } f(x) = y \text{ then } f(T(x)) = y$$

- A function f is **covariant** when it changes in a way consistent with the transformation T:

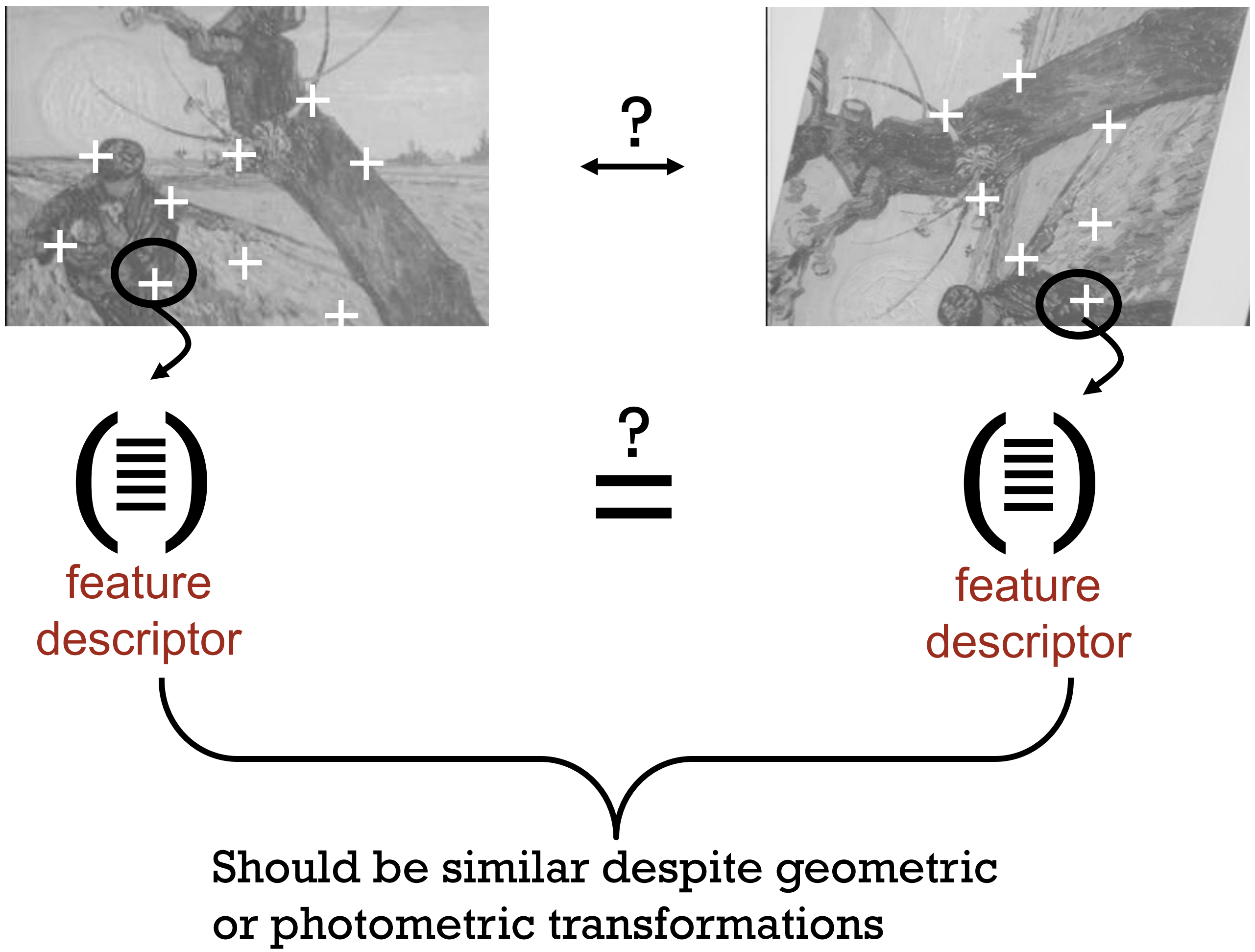
$$\text{if } f(x) = y \text{ then } f(T(x))=T(f(x))=T(y)$$

INTEREST POINT DETECTORS SHOULD BE COVARIANT

- Features should be detected in corresponding locations despite geometric or photometric changes.



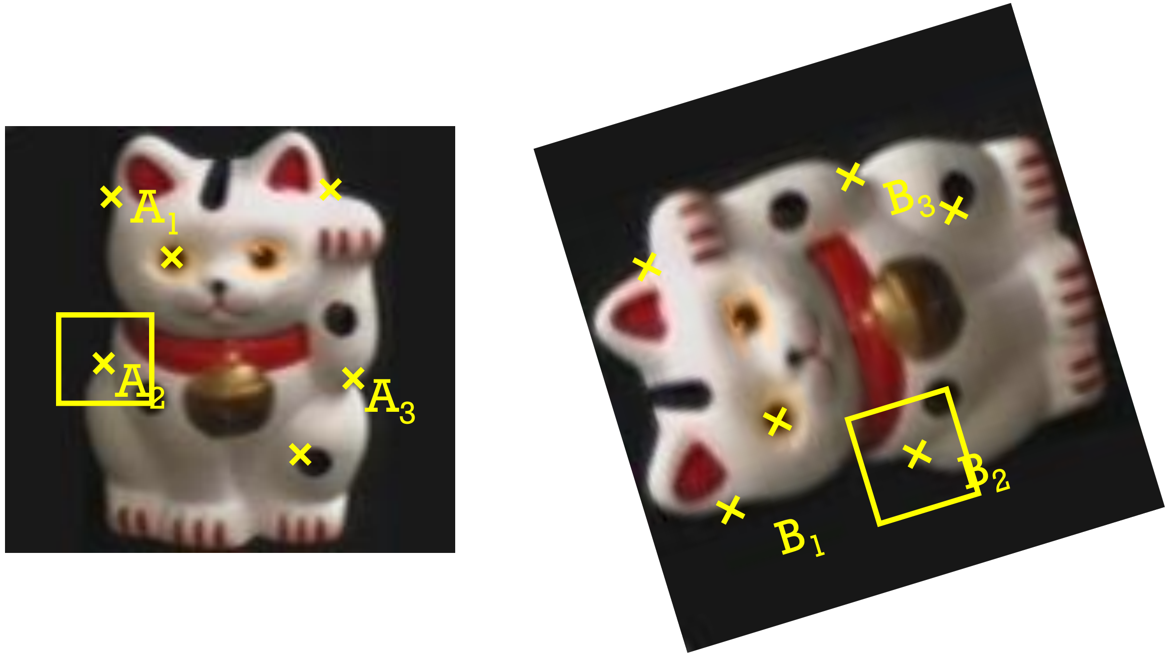
INTEREST POINT DESCRIPTORS SHOULD BE INVARIANT



OVERVIEW OF KEYPOINT MATCHING

1. **Find a set of distinctive key-points**

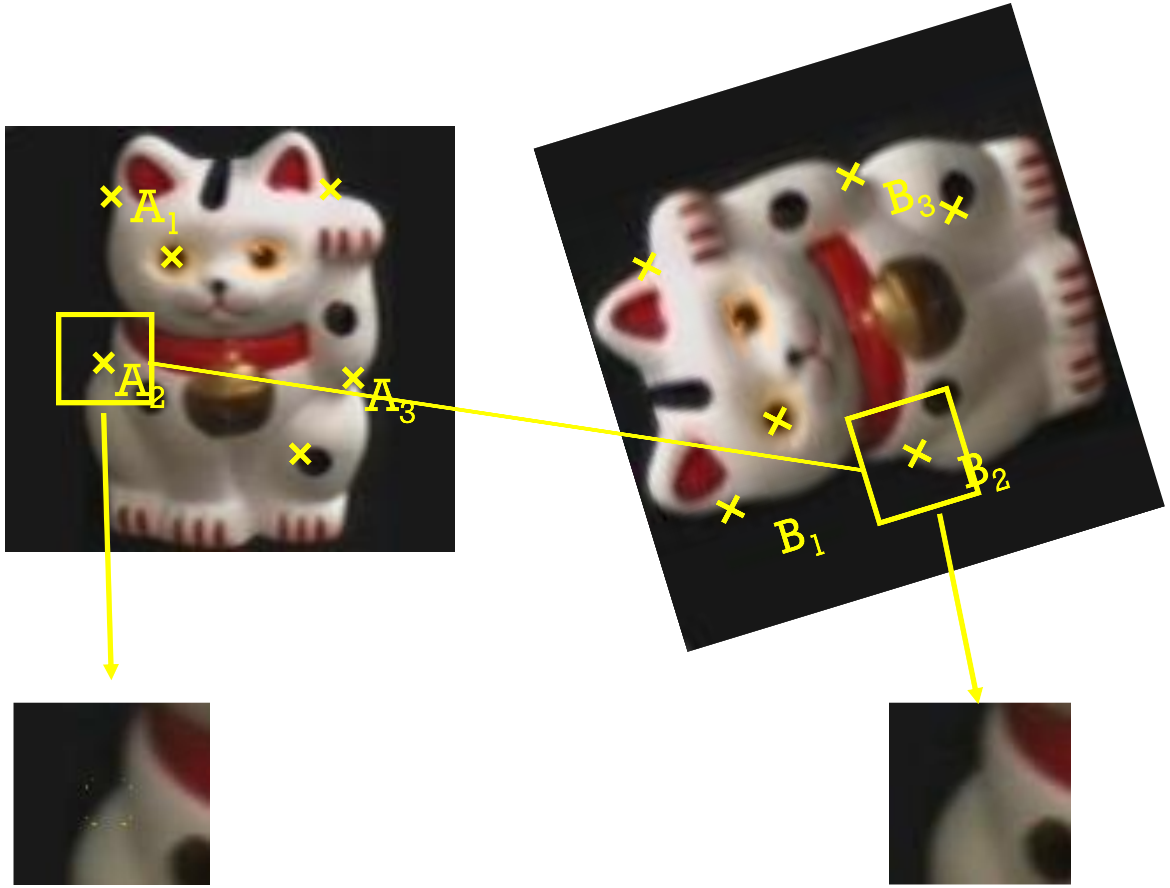
OVERVIEW OF KEYPOINT MATCHING



1. **Find a set of distinctive key-points**
2. **Define a region around each keypoint**

K. Grauman, B. Leibe

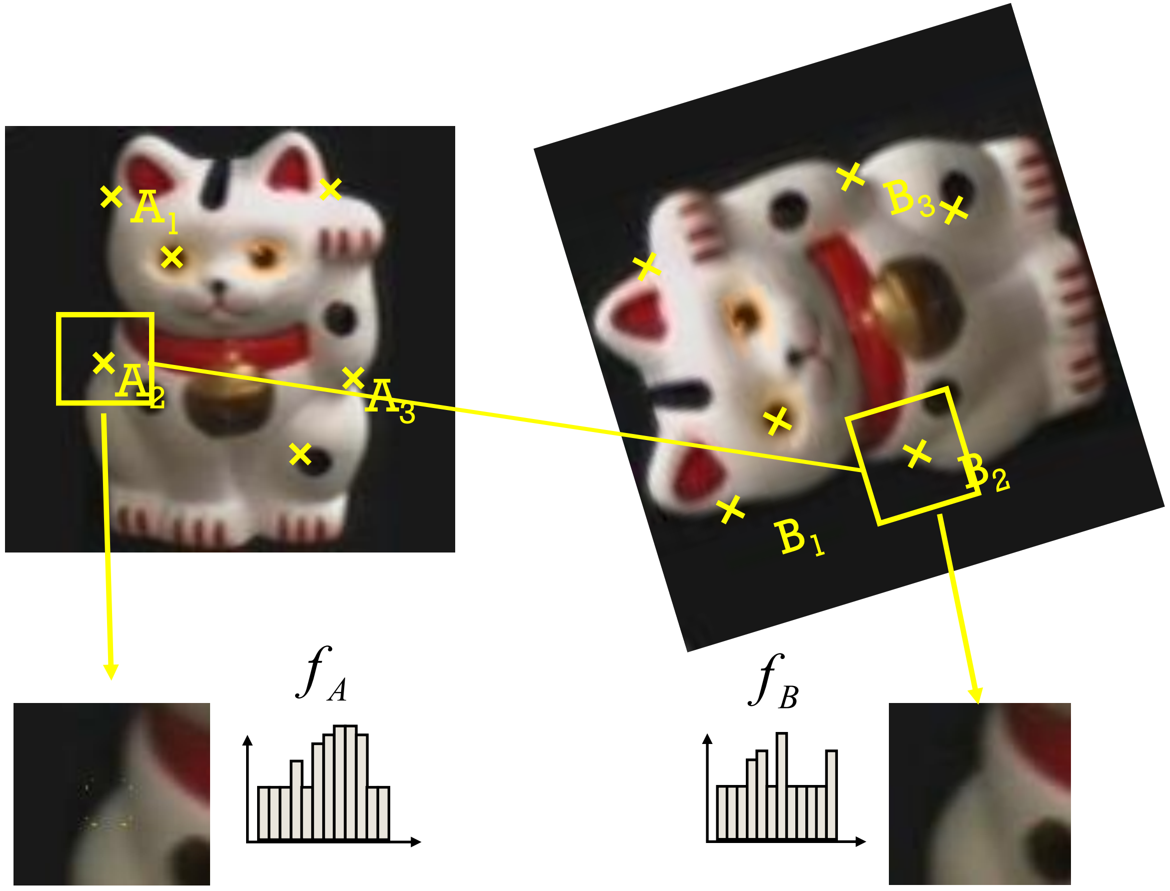
OVERVIEW OF KEYPOINT MATCHING



1. **Find a set of distinctive key-points**
2. **Define a region around each keypoint**
3. **Extract and normalize the region content**

K. Grauman, B. Leibe

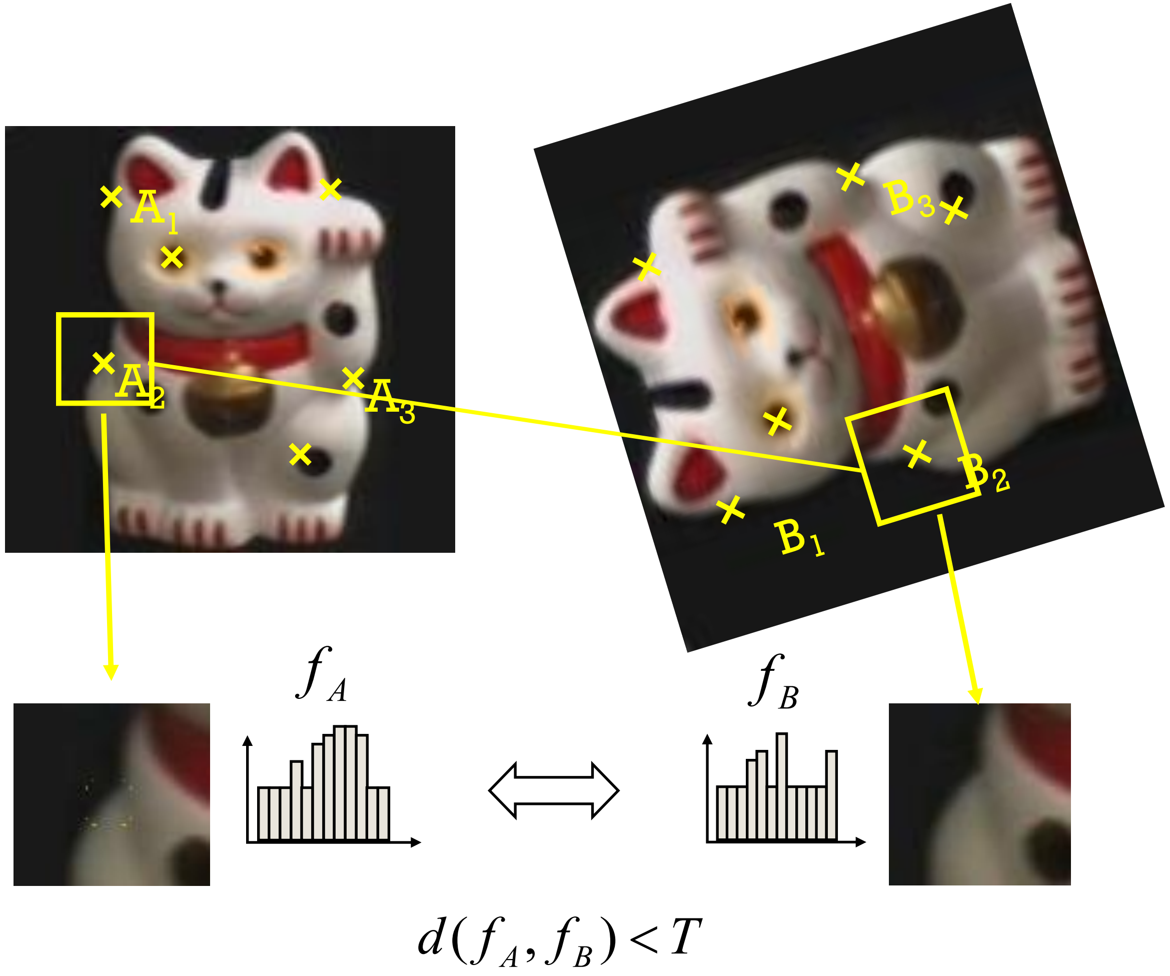
OVERVIEW OF KEYPOINT MATCHING



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

K. Grauman, B. Leibe

OVERVIEW OF KEYPOINT MATCHING



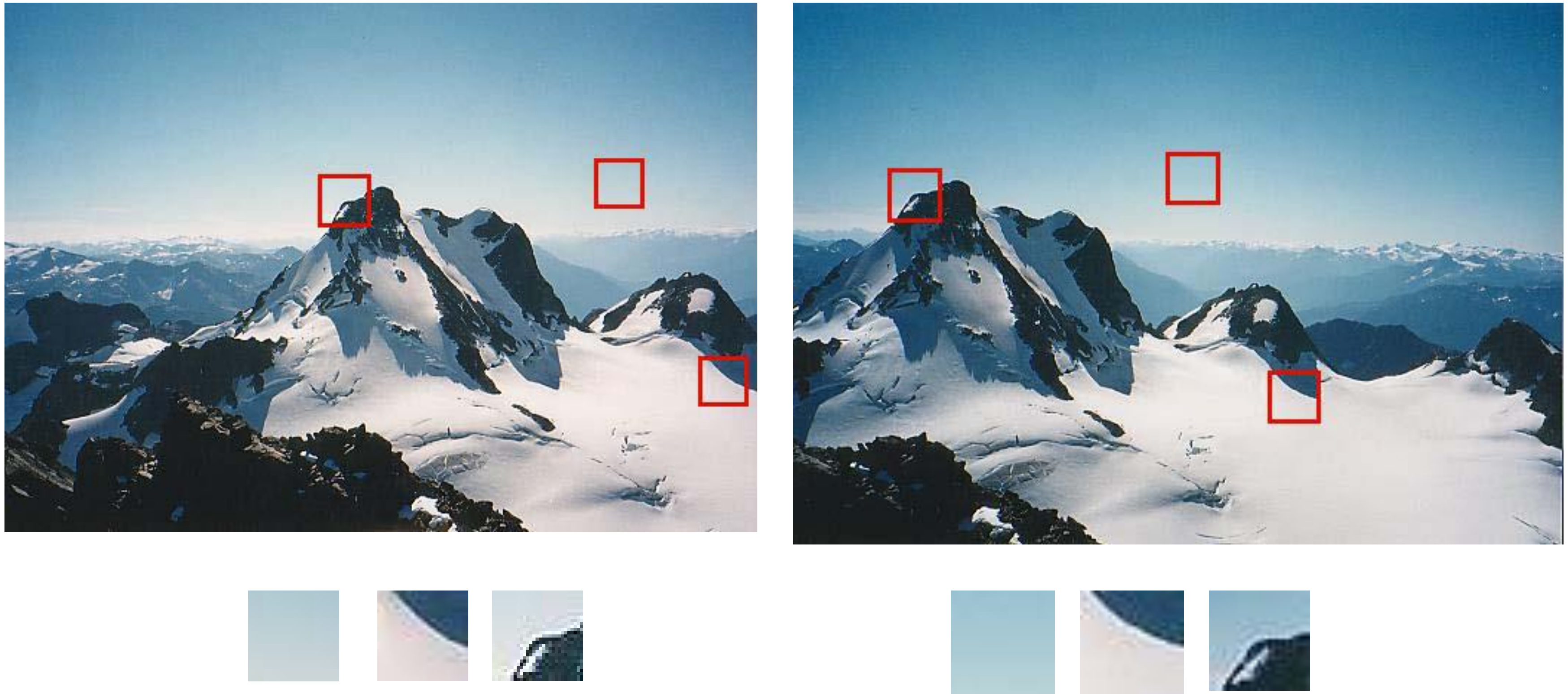
1. **Find a set of distinctive key-points**
2. **Define a region around each keypoint**
3. **Extract and normalize the region content**
4. **Compute a local descriptor from the normalized region**
5. **Match local descriptors**

K. Grauman, B. Leibe

GOALS FOR KEYPOINTS

Detect points that are *repeatable* and *distinctive*

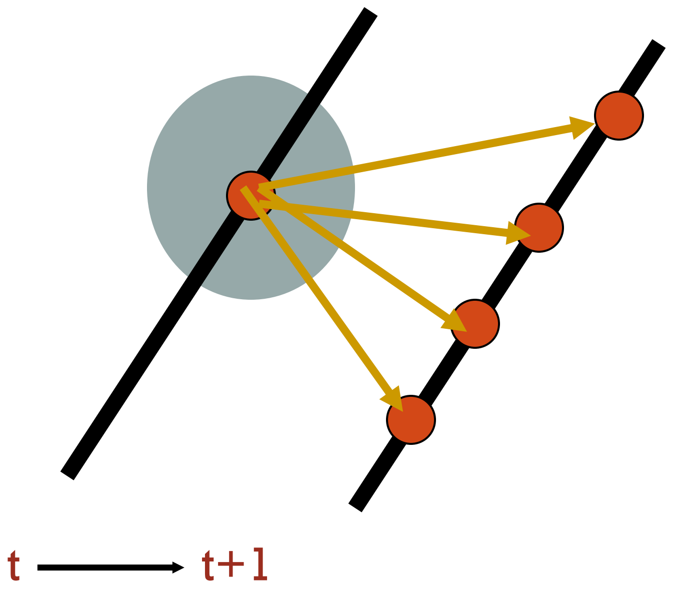
INTEREST POINT CANDIDATES



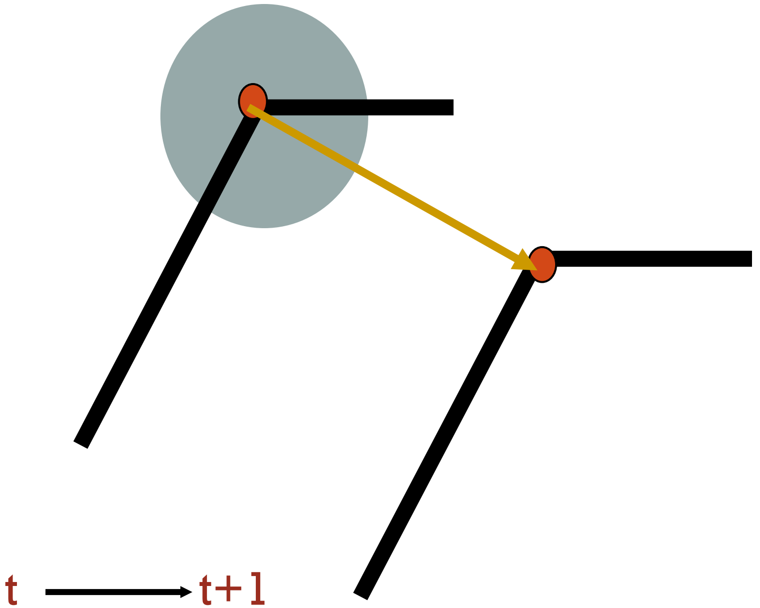
Use features with gradients in at least two, significantly different orientations (e.g., corners, junctions etc)

APERTURE PROBLEM

A point on a line is hard to match.

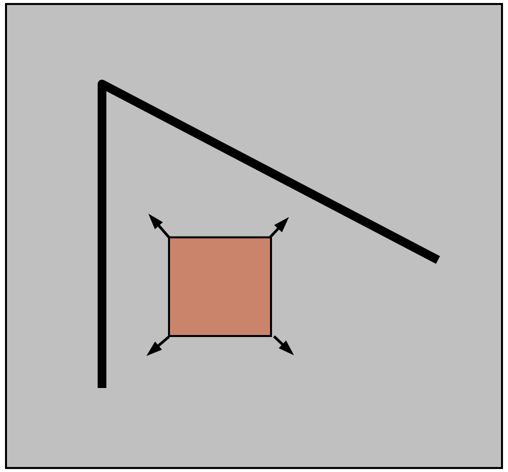


A corner is easier to match

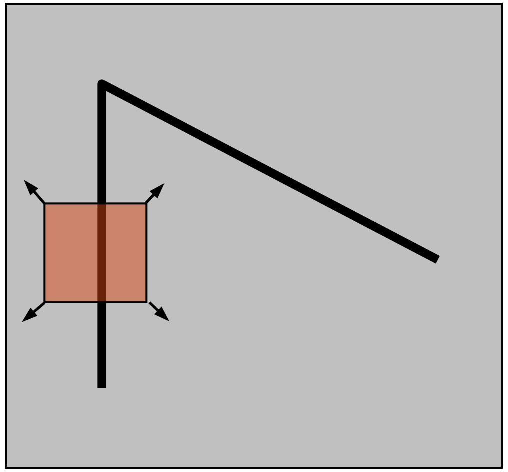


CORNER DETECTION: BASIC IDEA

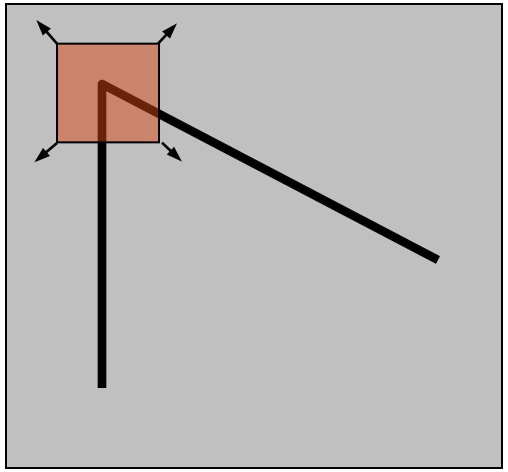
- How does the window change when you shift it?
- Shifting the window in any direction causes a big change



"flat" region:
no change in all directions



"edge":
no change along the edge direction

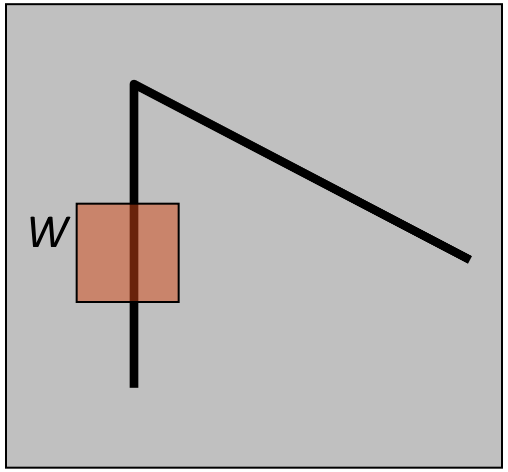


"corner":
significant change in all directions

Credit: S. Seitz, D. Frolova, D. Simakov

CORNER DETECTION: THE MATH

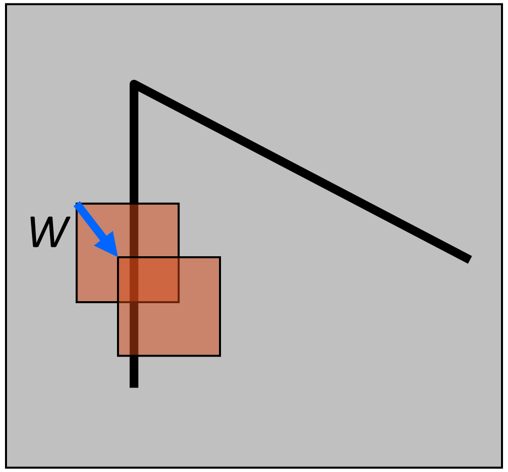
Consider shifting the window W by (u,v)



Corner Detection: The Math

Consider shifting the window *W* by (*u*,*v*)

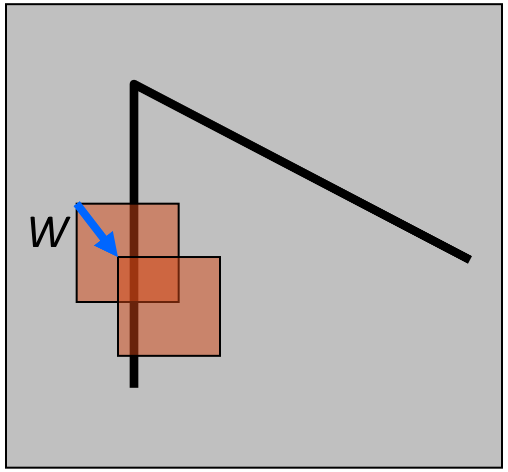
- how do the pixels in W change?



CORNER DETECTION: THE MATH

Consider shifting the window *W* by (*u*,*v*)

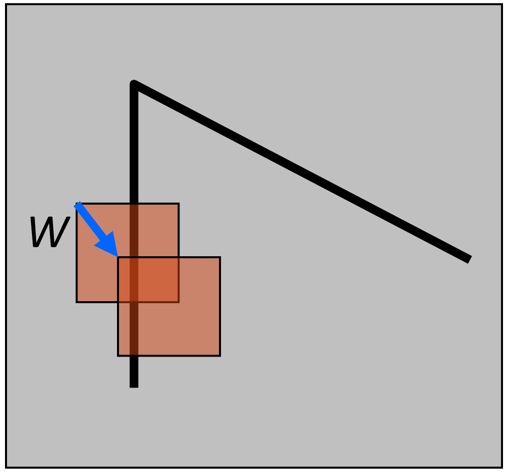
- how do the pixels in W change?
- compare each pixel before and after by summing up the squared differences (SSD)
- this defines an SSD "error" *E*(*u*,*v*):



CORNER DETECTION: THE MATH

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*):



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

SMALL MOTION ASSUMPTION

Taylor Series expansion of *I*:

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

SMALL MOTION ASSUMPTION

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, then first order approximation is good

$$I(x+u, y+v) \approx I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

SMALL MOTION ASSUMPTION

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

SMALL MOTION ASSUMPTION

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, then first order approximation is good

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

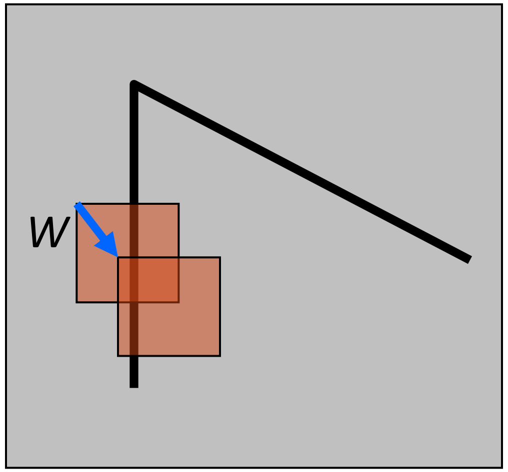
shorthand: $I_x = \frac{\partial I}{\partial x}$

Plugging this into the formula on the previous slide…

Corner Detection: the Math

Using the small motion assumption,
replace I with a linear approximation

(Shorthand: $I_x = \frac{\partial I}{\partial x}$)

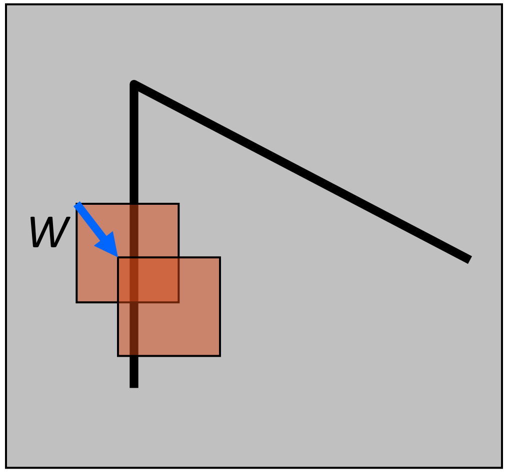


$$E(u,v) = \sum_{(x,y)\in W} \left(I(x+u, y+v) - I(x,y)\right)^2$$

CORNER DETECTION: THE MATH

Using the small motion assumption,
replace I with a linear approximation

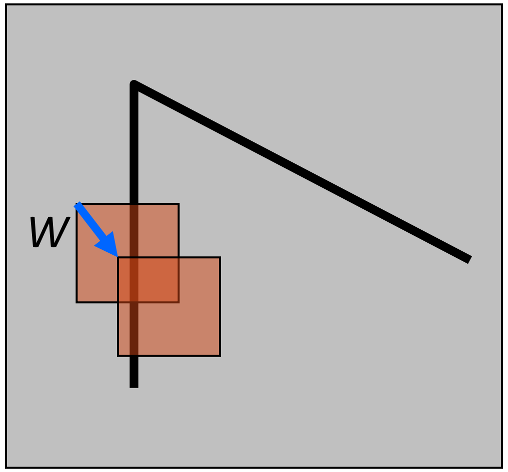
(Shorthand: $I_x = \frac{\partial I}{\partial x}$)



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

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

CORNER DETECTION: THE MATH



Using the small motion assumption, replace I with a linear approximation

(Shorthand: $I_x = \frac{\partial I}{\partial x}$)

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

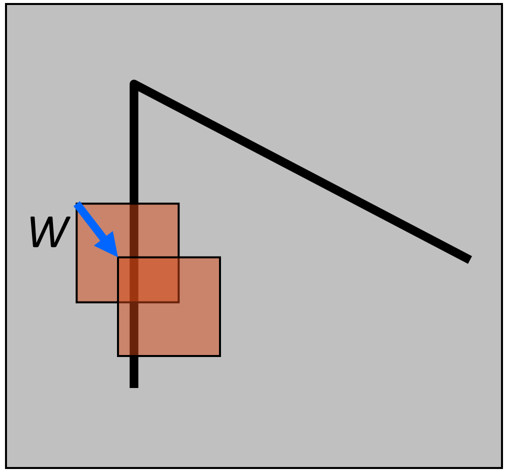
$$\approx \sum_{(x,y)\in W} \left(I(x,y) + I_x(x,y)u + I_y(x,y)v - I(x,y)\right)^2$$

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

Corner Detection: The Math

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

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



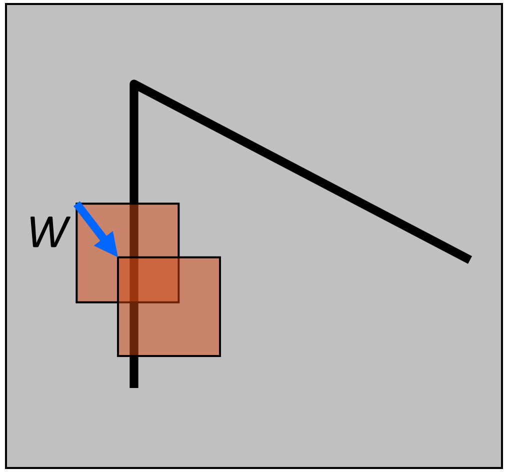
CORNER DETECTION: THE MATH

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

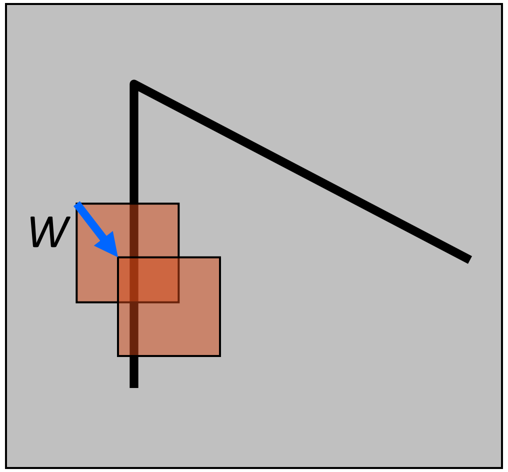
$$\approx \sum_{(x,y)\in W} (I_x^2u^2 + 2I_xI_yuv + I_y^2v^2)$$

$$\approx \quad Au^2 + 2Buv + Cv^2$$

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



CORNER DETECTION: THE MATH



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

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

$$\approx Au^2 + 2Buv + Cv^2$$

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

- Thus, *E*(*u*,*v*) is locally approximated as a *quadratic form*

THE SECOND MOMENT MATRIX

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

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

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

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

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

THE SECOND MOMENT MATRIX

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

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

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

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

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

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

THE SECOND MOMENT MATRIX

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

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

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

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

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

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

THE SECOND MOMENT MATRIX

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

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

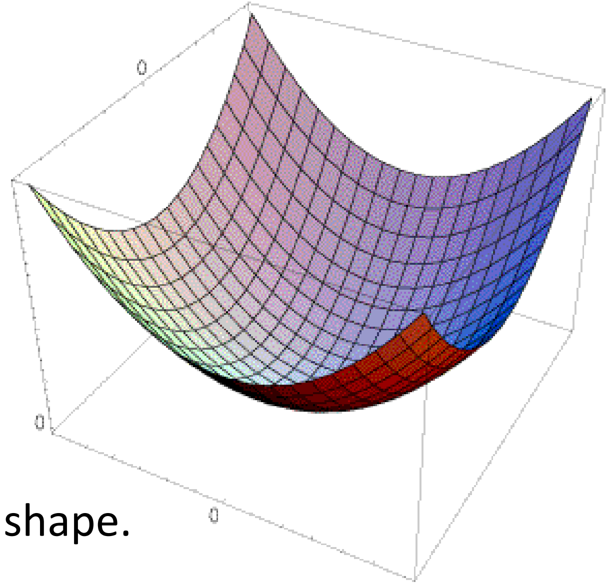
$$\approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

H is a 2 x 2 matrix called auto-correlation or 2nd order moment matrix

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

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

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



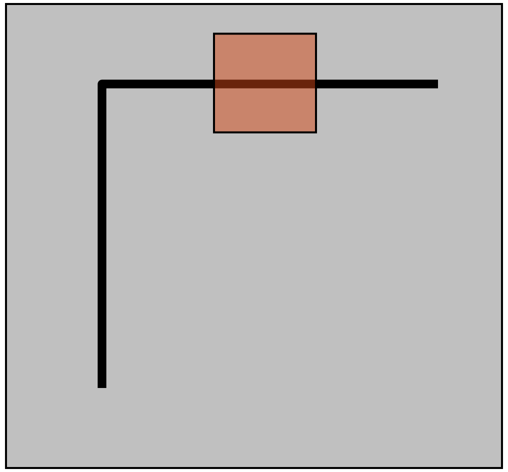
Let's try to understand its shape.

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

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

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

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



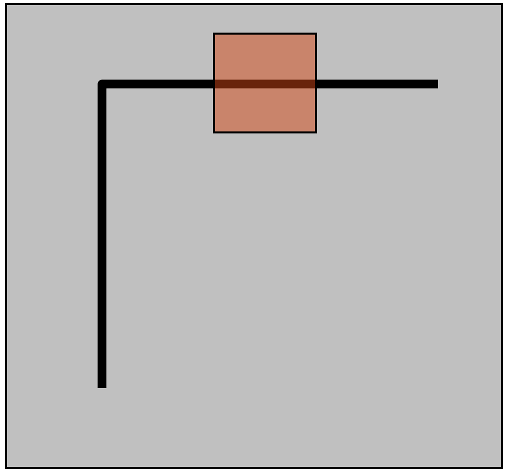
Horizontal edge: $I_x = 0$

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

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

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

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



Horizontal edge: $I_x = 0$

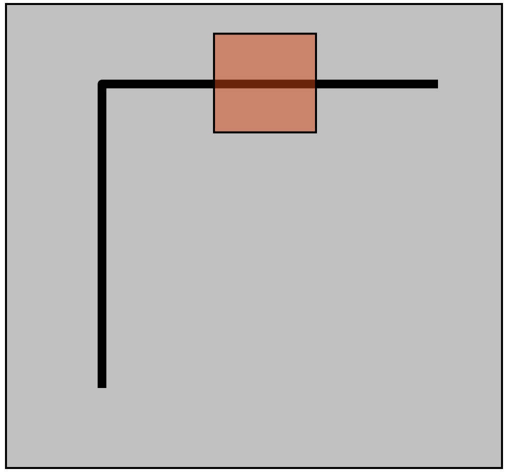
$$H = \begin{bmatrix} 0 & 0 \\ 0 & C \end{bmatrix}$$

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

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

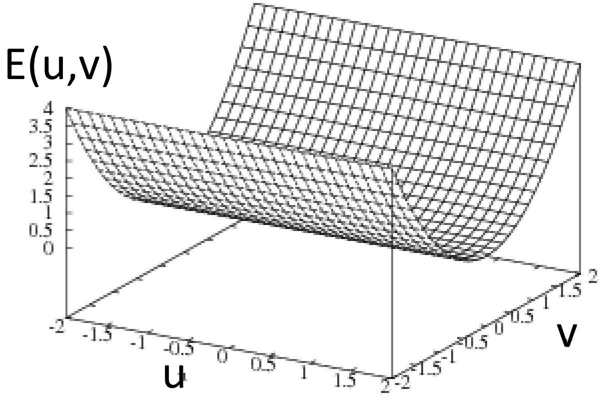
$$B = \sum_{(x,y)\in W} I_x I_y$$

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



Horizontal edge: $I_x = 0$

$$H = \begin{bmatrix} 0 & 0 \\ 0 & C \end{bmatrix}$$

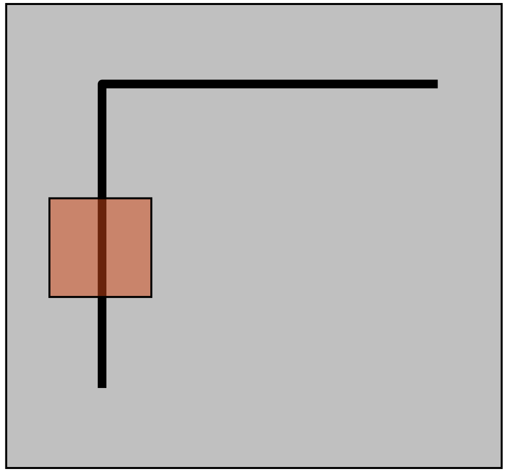


$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

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

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

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



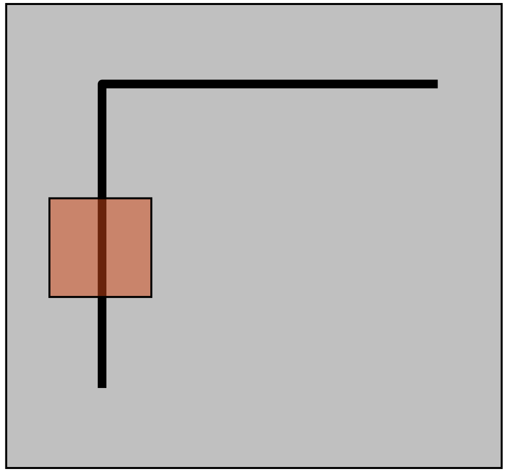
Vertical edge: $I_y = 0$

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

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

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

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



Vertical edge: $I_y = 0$

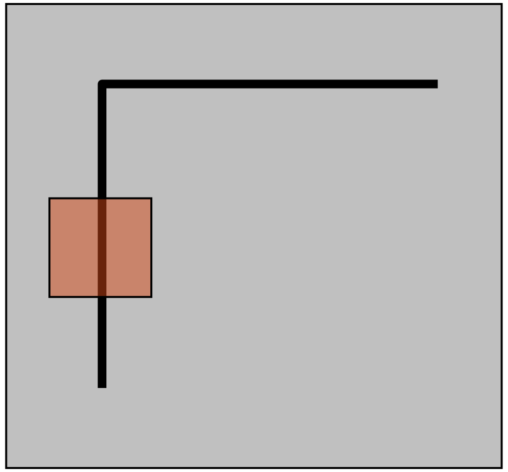
$$H = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}$$

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

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

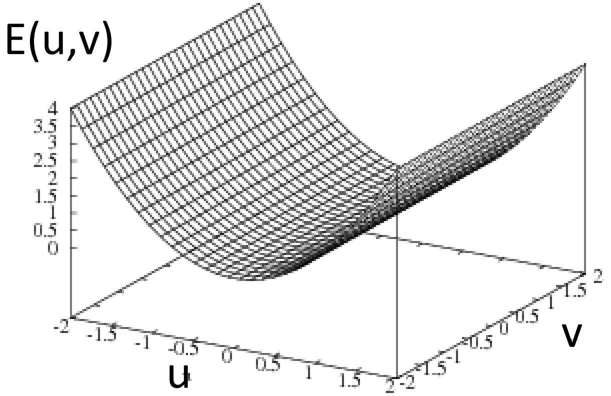
$$B = \sum_{(x,y)\in W} I_x I_y$$

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

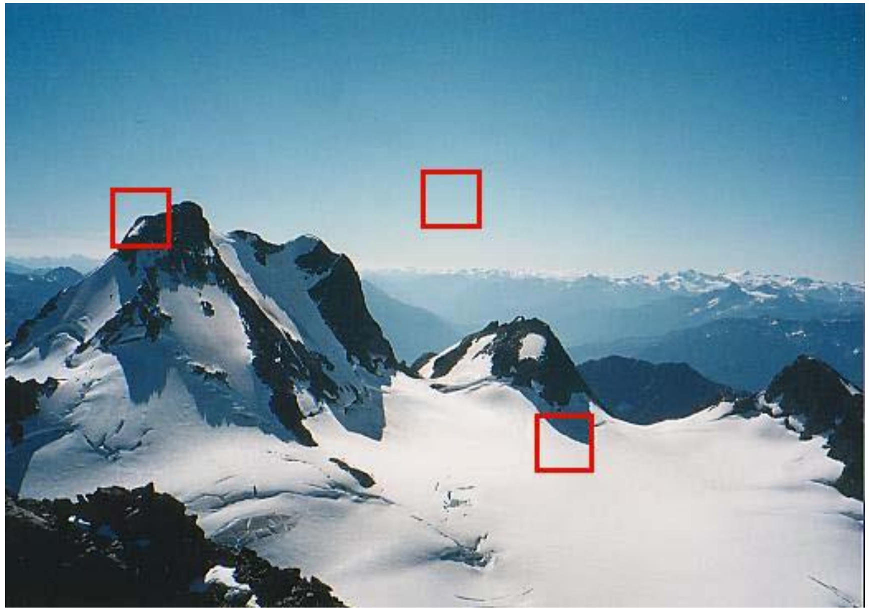


Vertical edge: $I_y = 0$

$$H = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}$$

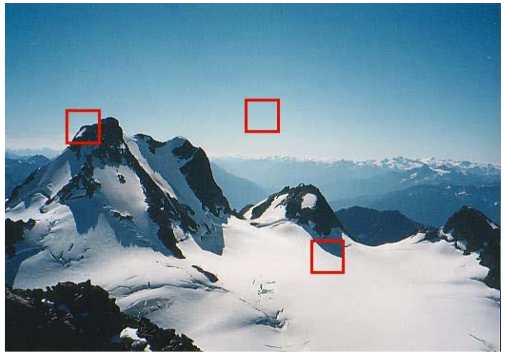


PROPERTIES OF AUTO-CORRELATION MATRIX



Describes the gradient distribution
(i.e., local structure) inside the window!

GENERAL CASE

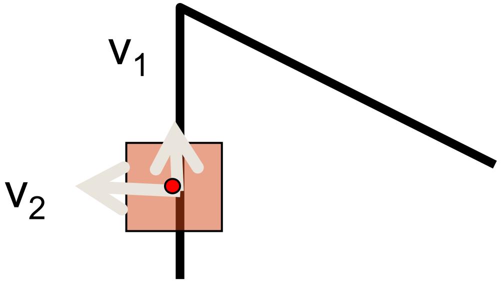


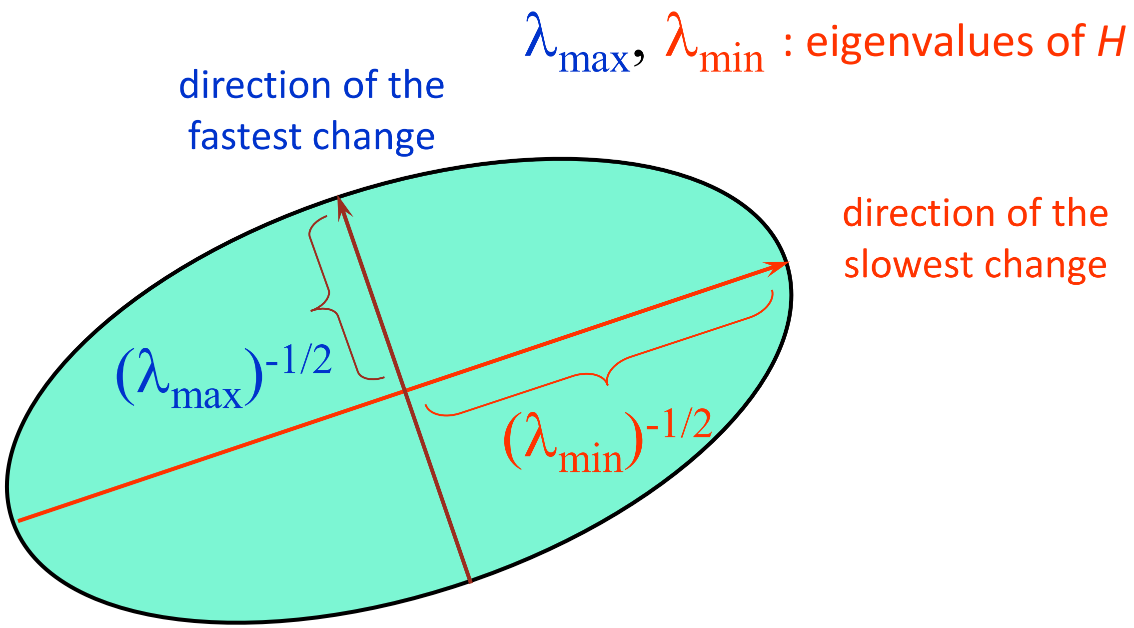
The shape of H tells us something about the *distribution of gradients* around a pixel

We can visualize H as an ellipse with axis lengths determined by the *eigenvalues* of H and orientation determined by the *eigenvectors* of H

Ellipse equation:

$$[u \;\; v] \;\; H \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$





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**

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:

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

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

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

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:

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

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

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

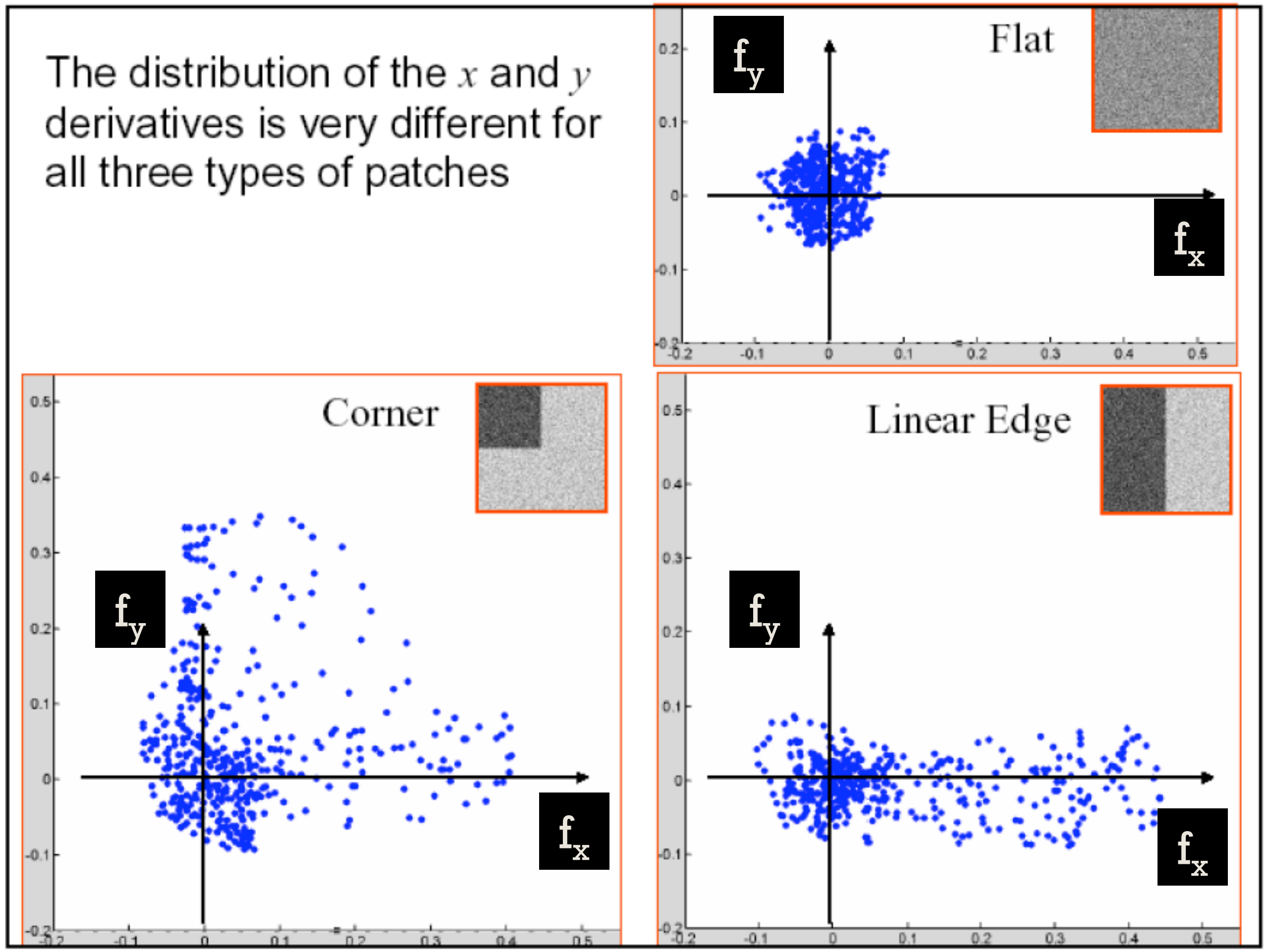
- The solution:

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

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

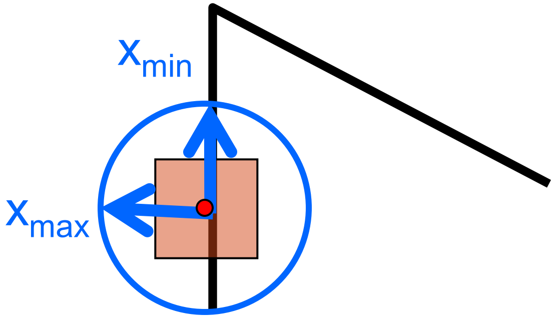
$$\begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0$$

DISTRIBUTION OF FX AND FY



Corner Detection: The Math

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



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

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

Eigenvalues and eigenvectors 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}

CORNER DETECTION: THE MATH

How are λ_{max}, x_{max}, λ_{min}, and x_{min} relevant for feature detection?

- What's our feature scoring function?

CORNER DETECTION: THE MATH

How are λ_{max}, x_{max}, λ_{min}, and x_{min} relevant for feature detection?

- What's our feature scoring function?

Want $E(u,v)$ to be large for small shifts in all directions

- the minimum of $E(u,v)$ should be large, over all unit vectors [u v]
- this minimum is given by the smaller eigenvalue (λ_{min}) of H

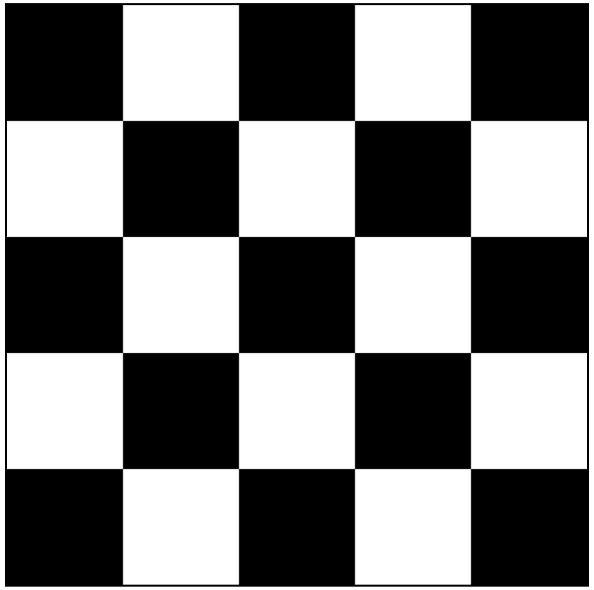
CORNER DETECTION: THE MATH

How are λ_{max}, x_{max}, λ_{min}, and x_{min} relevant for feature detection?

- What's our feature scoring function?

Want $E(u,v)$ to be large for small shifts in all directions

- the minimum of $E(u,v)$ should be large, over all unit vectors $[u\ v]$
- this minimum is given by the smaller eigenvalue (λ_{min}) of H



I

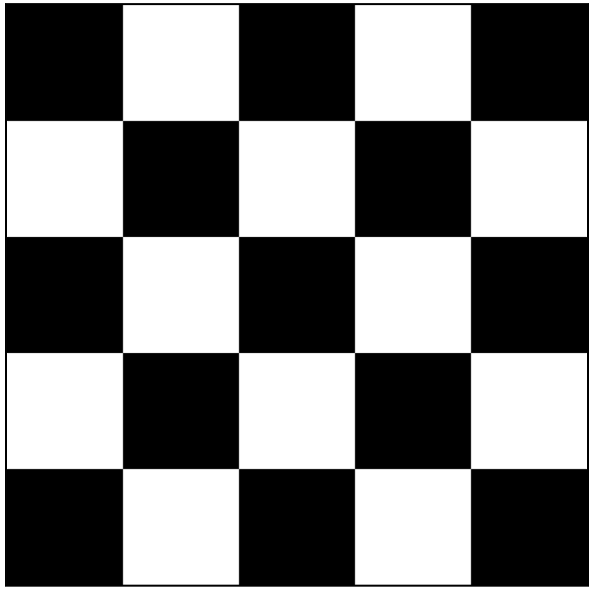
CORNER DETECTION: THE MATH

How are λ_{max}, x_{max}, λ_{min}, and x_{min} relevant for feature detection?

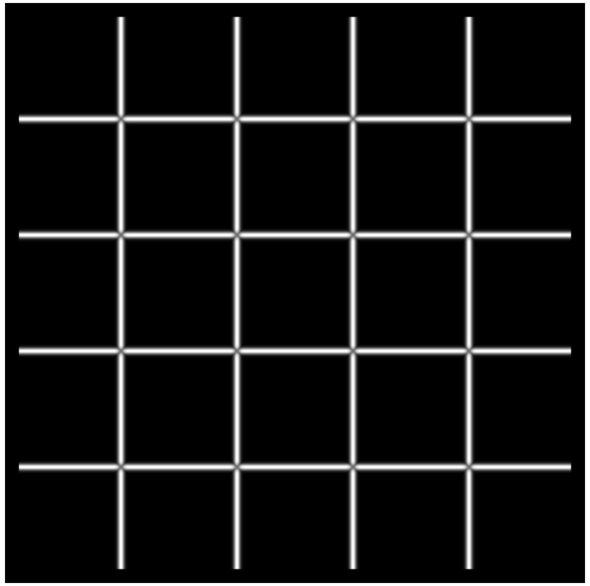
- What's our feature scoring function?

Want $E(u,v)$ to be large for small shifts in all directions

- the minimum of $E(u,v)$ should be large, over all unit vectors $[u\ v]$
- this minimum is given by the smaller eigenvalue (λ_{min}) of H



I



$\lambda_{\max}$

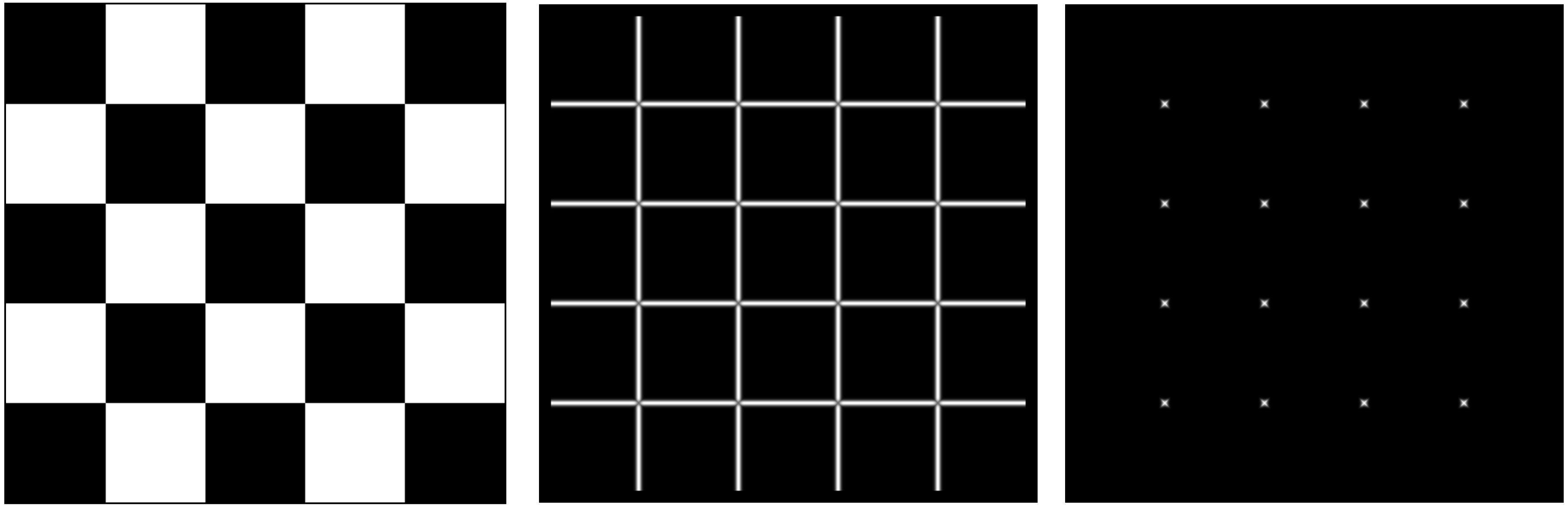
CORNER DETECTION: THE MATH

How are λ_{max}, x_{max}, λ_{min}, and x_{min} relevant for feature detection?

- What's our feature scoring function?

Want $E(u,v)$ to be large for small shifts in all directions

- the minimum of $E(u,v)$ should be large, over all unit vectors $[u\ v]$
- this minimum is given by the smaller eigenvalue (λ_{min}) of H

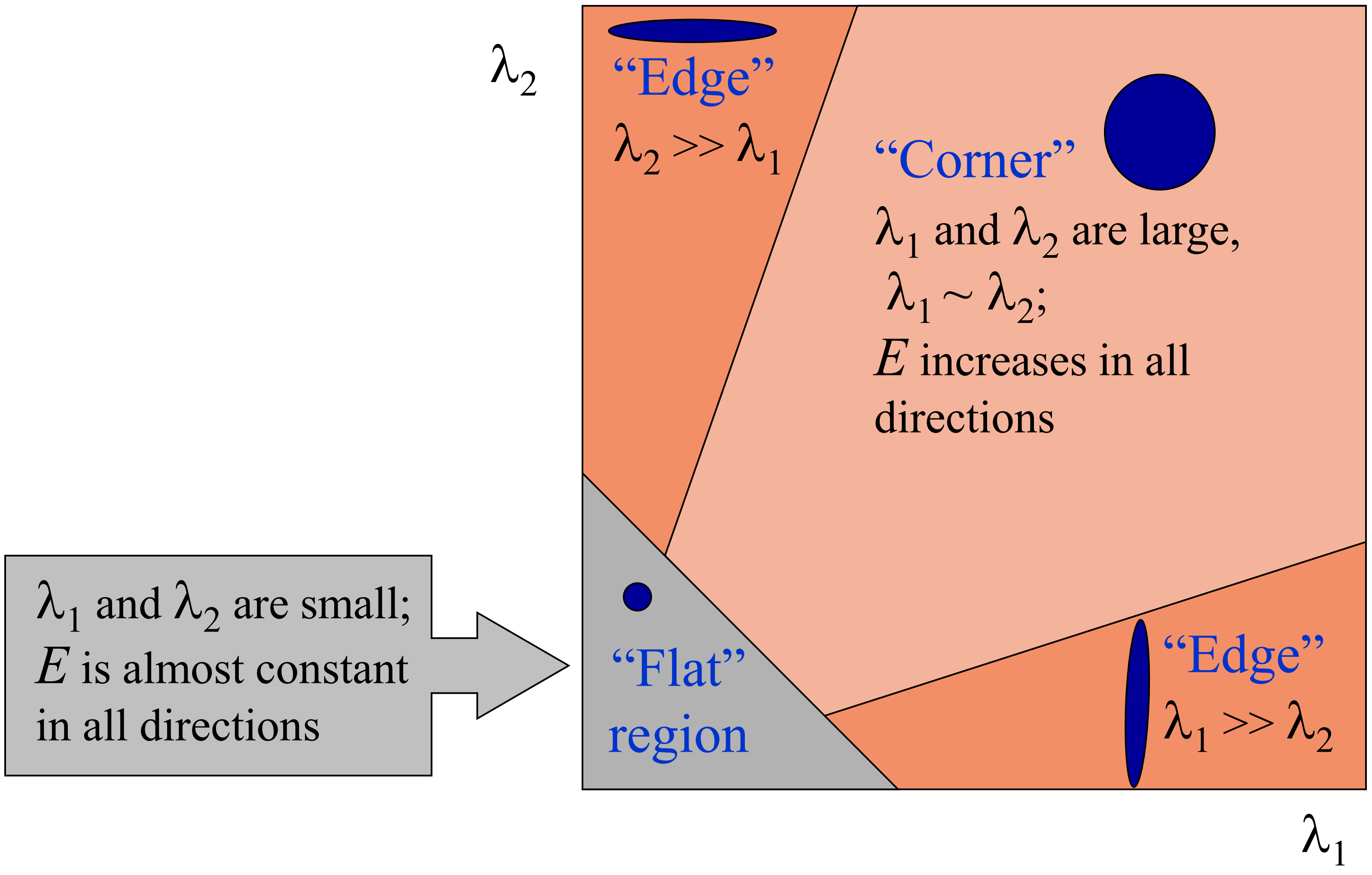


I $\quad$ λ_{max} $\quad$ λ_{min}

J. Shi and C. Tomasi (June 1994). "Good Features to Track,". *9th IEEE Conference on Computer Vision and Pattern Recognition*. Springer.

INTERPRETING THE EIGENVALUES

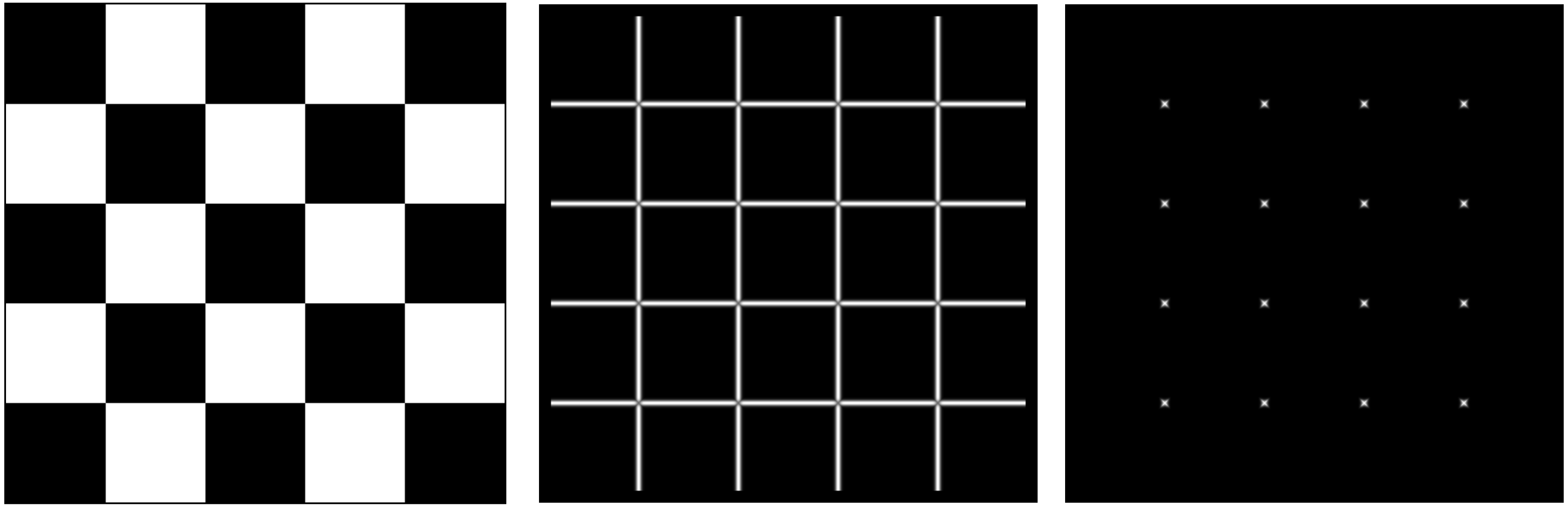
Classification of image points using eigenvalues of *M*:



CORNER DETECTION SUMMARY

Here's what you do

- Compute the gradient at each point in the image
- Create the *H* matrix from the entries in the gradient
- Compute the eigenvalues.
- Find points with large response (λ_{min} > threshold)
- Choose those points where λ_{min} is a local maximum as features



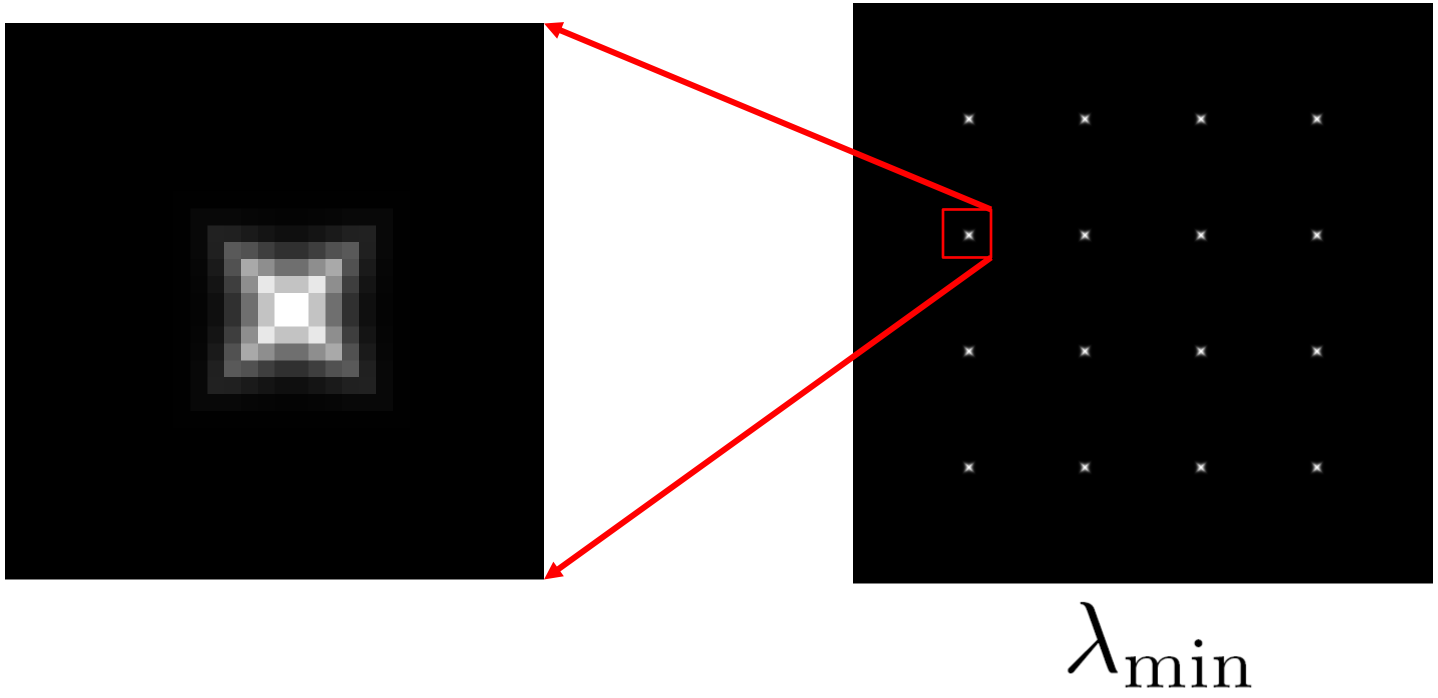
I λ_{max} λ_{min}

J. Shi and C. Tomasi (June 1994). "Good Features to Track,". *9th IEEE Conference on Computer Vision and Pattern Recognition*. Springer.

CORNER DETECTION SUMMARY

Here's what you do

- Compute the gradient at each point in the image
- Create the *H* matrix from the entries in the gradient
- Compute the eigenvalues.
- Find points with large response (λ_{min} > threshold)
- Choose those points where λ_{min} is a local maximum as features



J. Shi and C. Tomasi (June 1994). "Good Features to Track,". *9th IEEE Conference on Computer Vision and Pattern Recognition*. Springer.

THE HARRIS OPERATOR

λ_{min} is a variant of the "Harris operator"[1] for feature detection

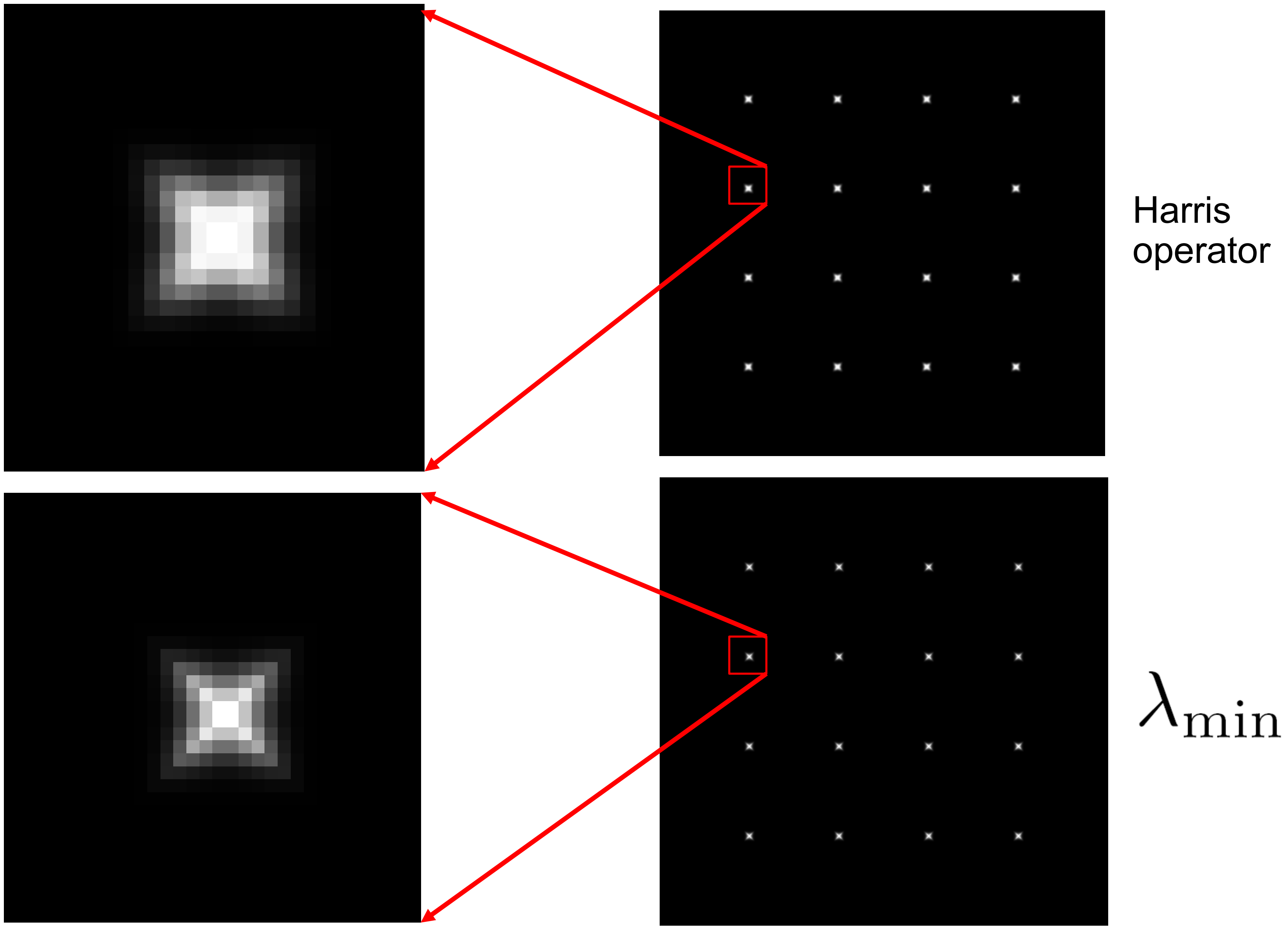
$$f = \lambda_1 \lambda_2 - \kappa (\lambda_1 + \lambda_2)^2$$

$$= determinant(H) - \kappa \left(trace(H)\right)^2$$

- The *trace* is the sum of the diagonals, i.e., ***trace(H)* = *h*$_{11}$ + *h*$_{22}$**
- Called the "Harris Corner Detector" or "Harris Operator"
- Lots of other detectors, this is one of the most popular

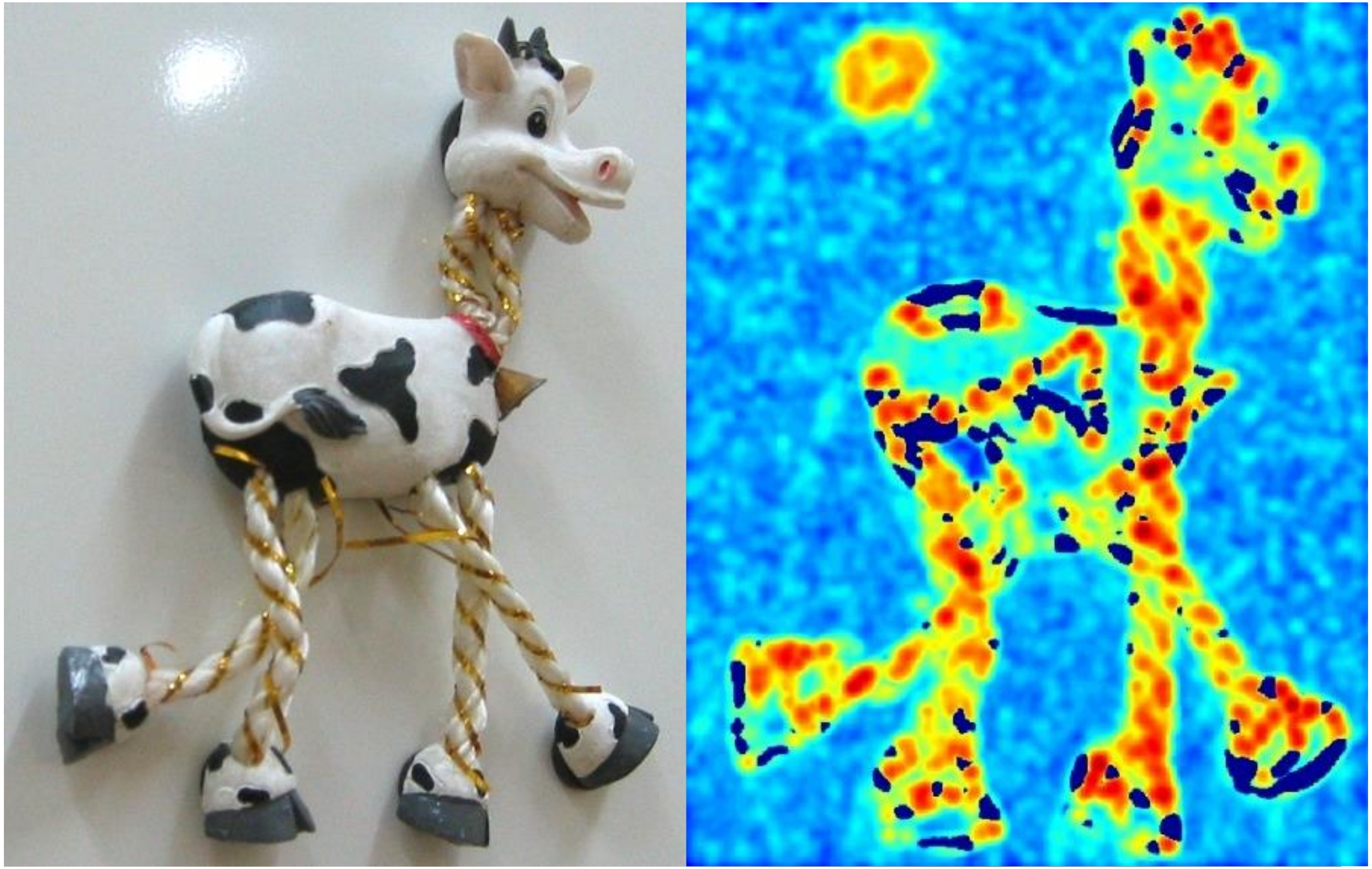
[1]C. Harris and M. Stephens (1988). "A combined corner and edge detector". *Proceedings of the 4th Alvey Vision Conference*. pp. 147–151.

THE HARRIS OPERATOR

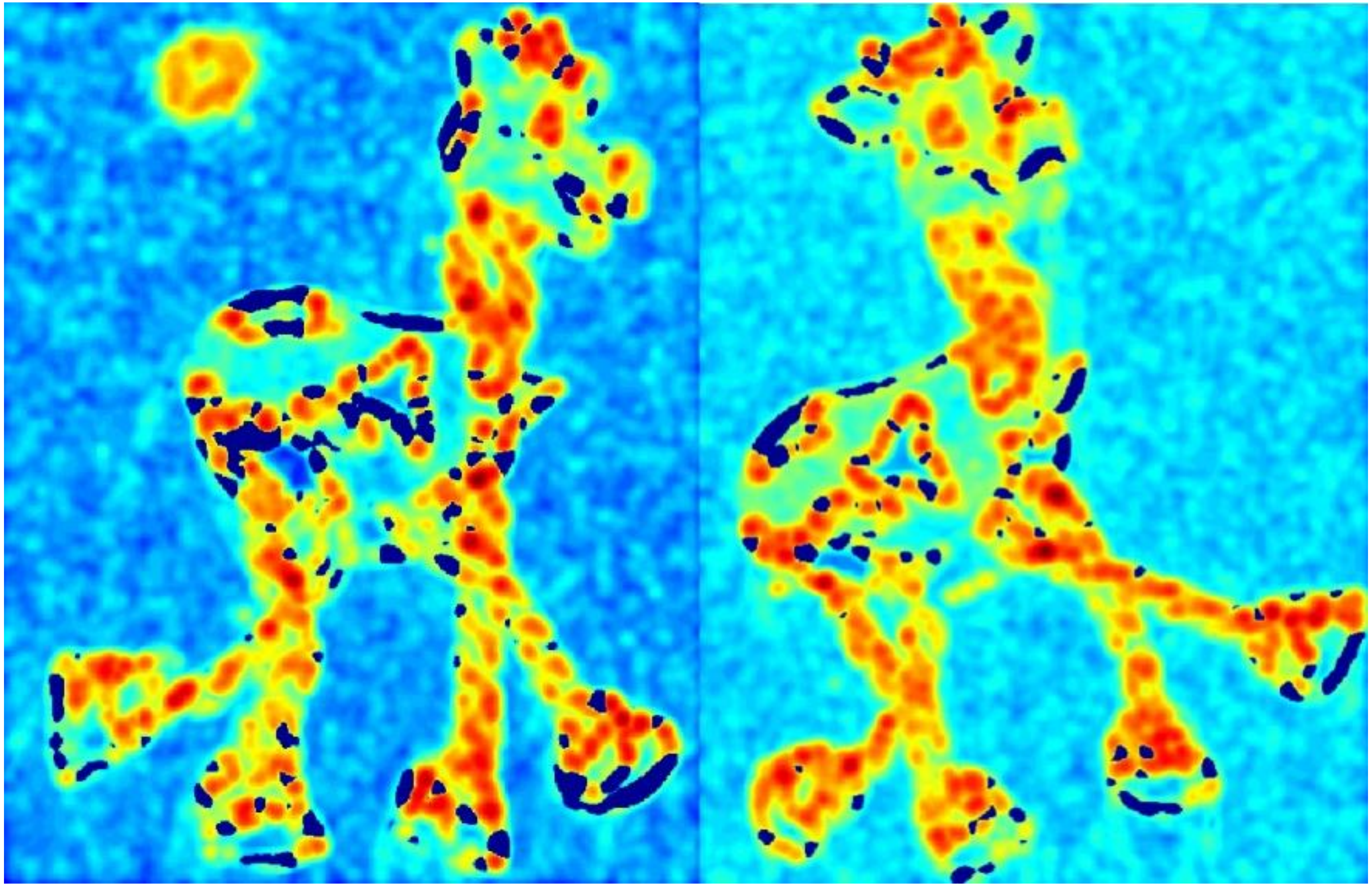


HARRIS DETECTOR EXAMPLE

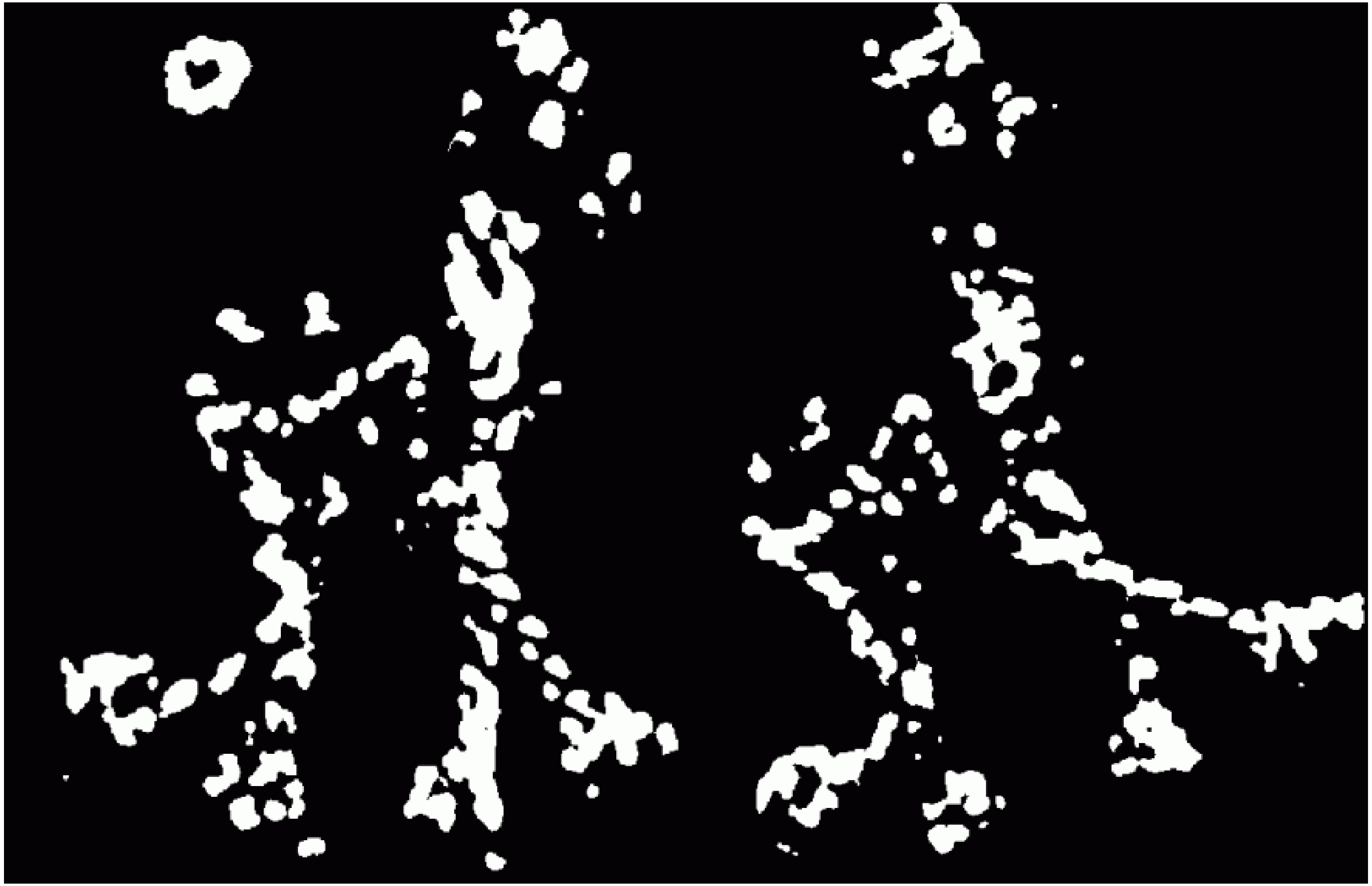
f value (red high, blue low)



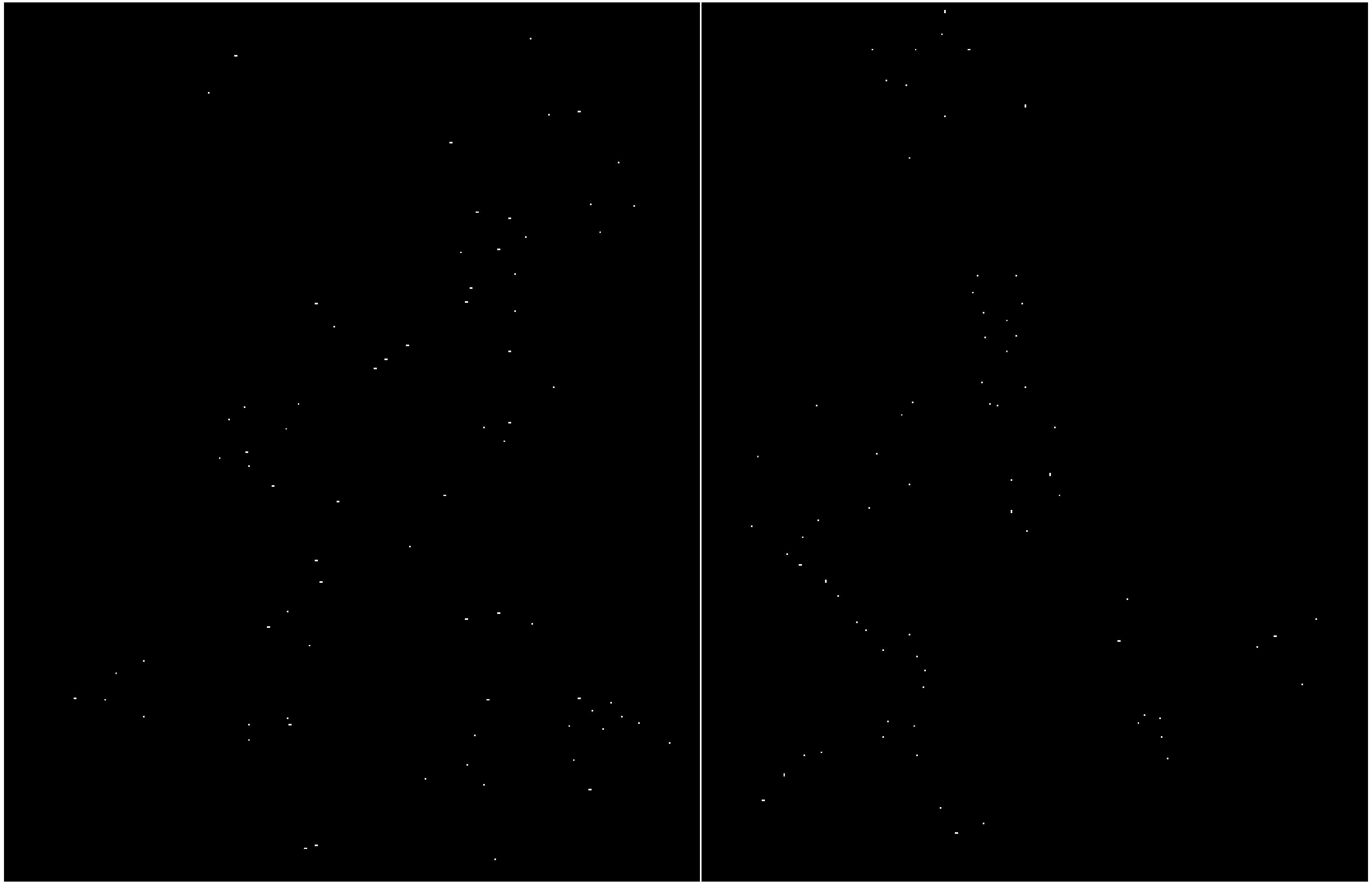
F VALUE (RED HIGH, BLUE LOW)



THRESHOLD (F > VALUE)



FIND LOCAL MAXIMA OF F



HARRIS FEATURES (IN RED)

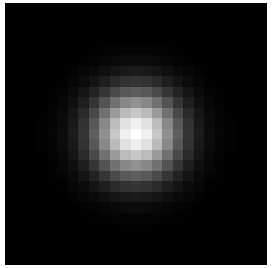
WEIGHTING THE DERIVATIVES

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

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

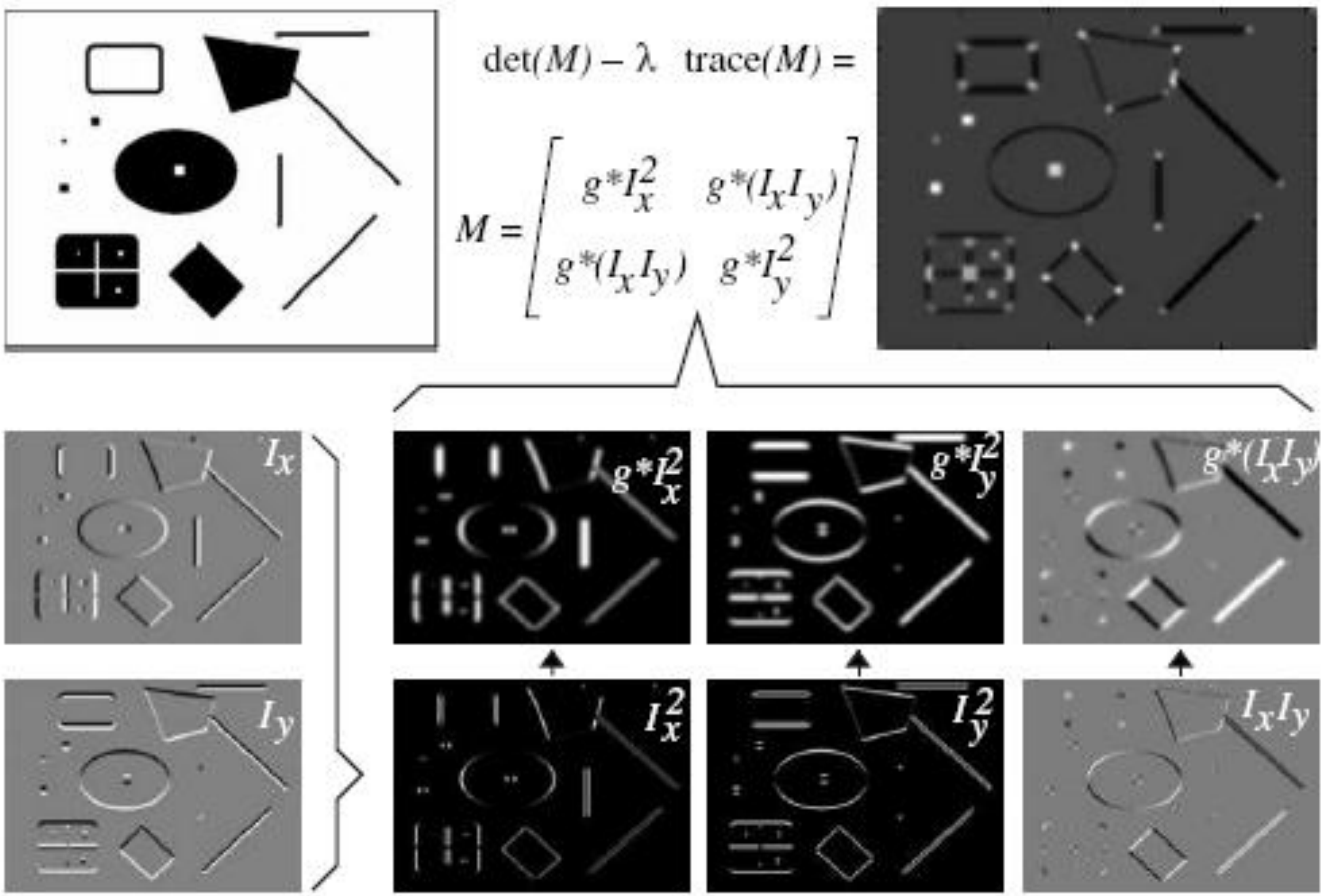
- Instead, we'll *weight* each derivative value based on its distance from the center pixel

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



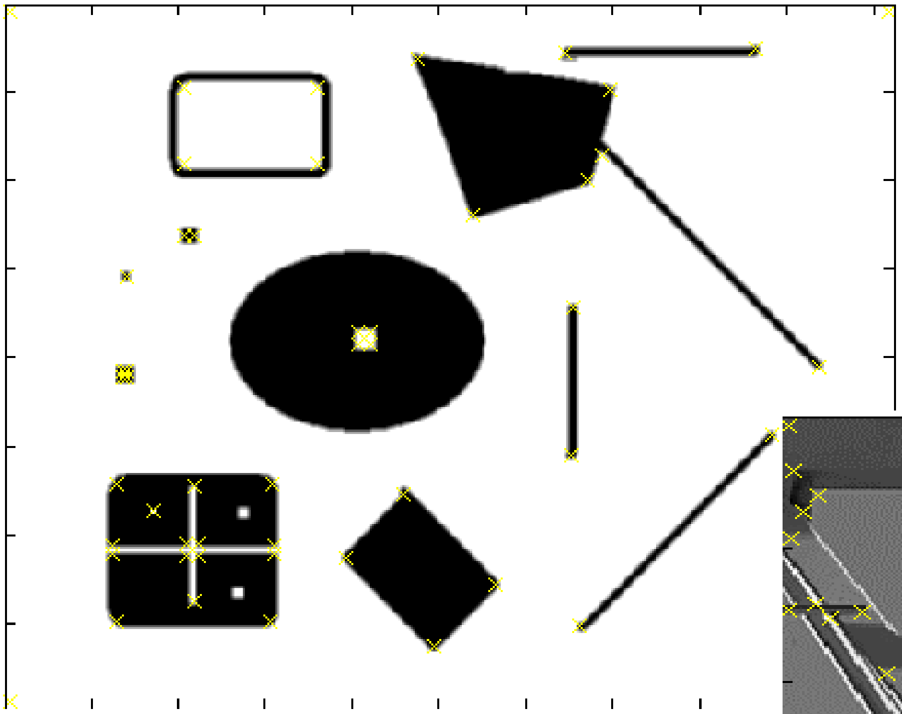
$w_{x,y}$

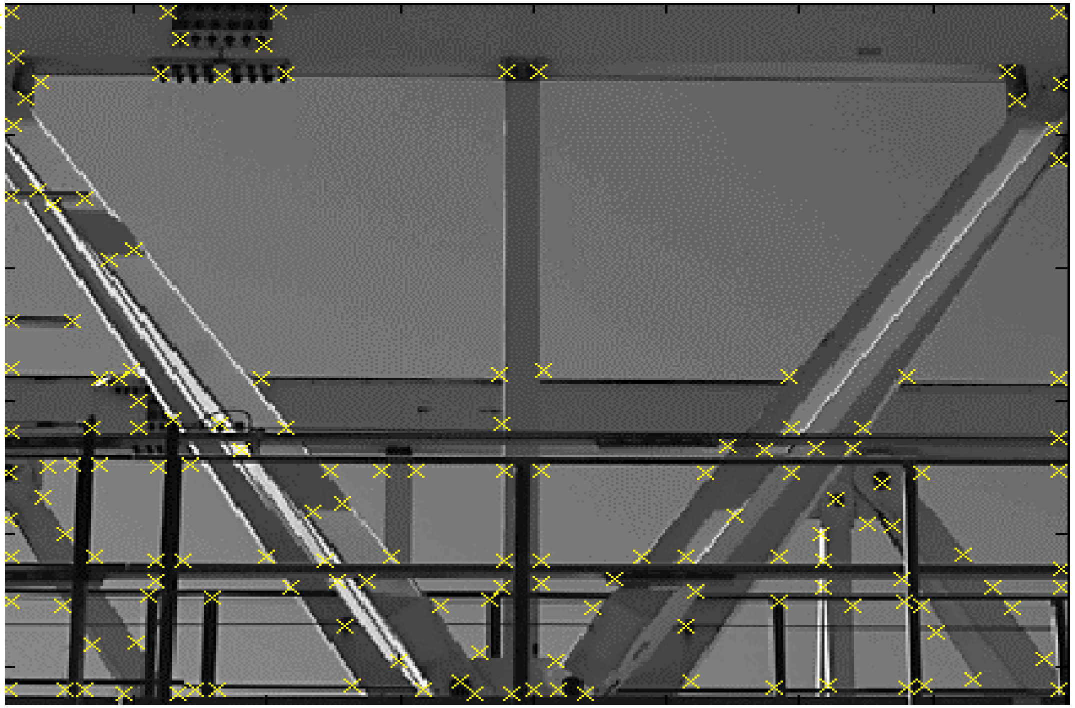
HARRIS DETECTOR – RESPONSES [HARRIS88]



HARRIS DETECTOR – RESPONSES

[HARRIS88]

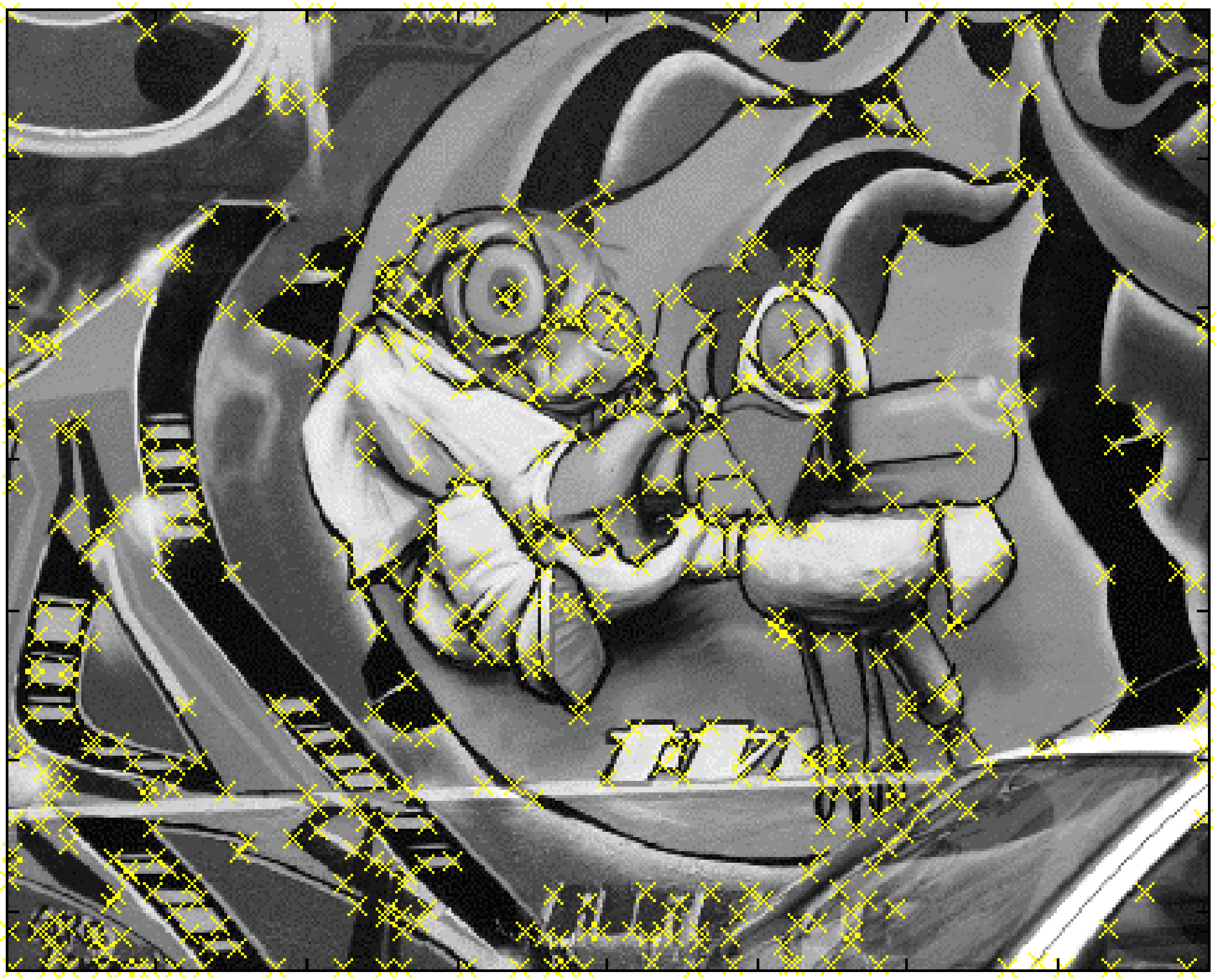




Effect: **A very precise corner detector.**

HARRIS DETECTOR – RESPONSES

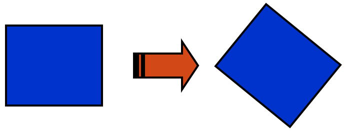
[HARRIS88]



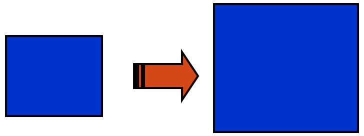
INVARIANCE TO GEOMETRIC/PHOTOMETRIC CHANGES

- Is the Harris detector invariant to geometric and photometric changes?

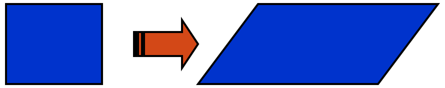
  - Rotation



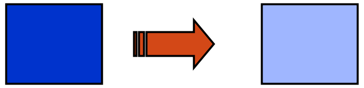
  - Scale



  - Affine

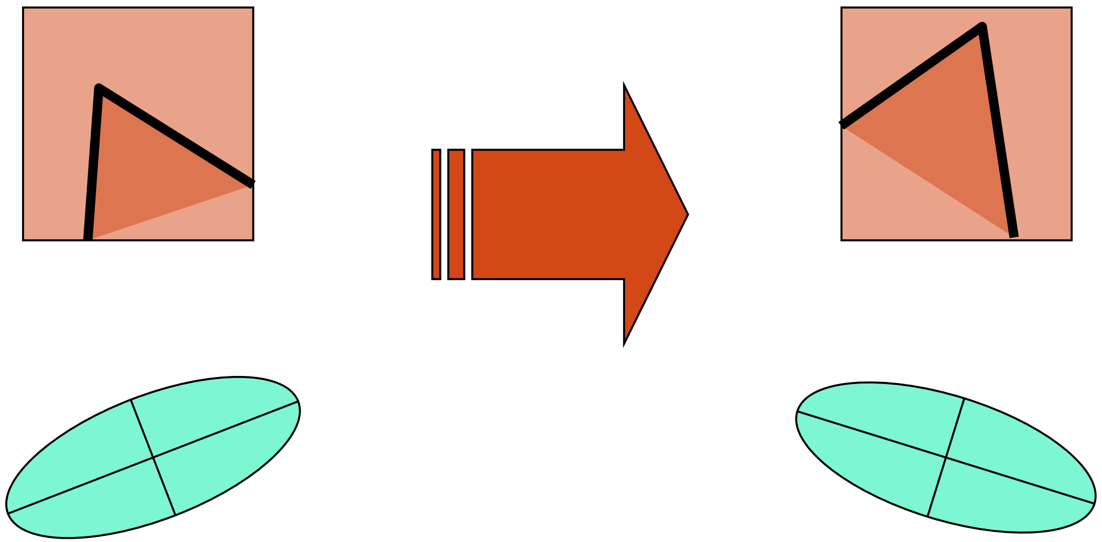


  - Intensity change: $I(x,y) \rightarrow a\, I(x,y) + b$



HARRIS DETECTOR: INVARIANCE PROPERTIES

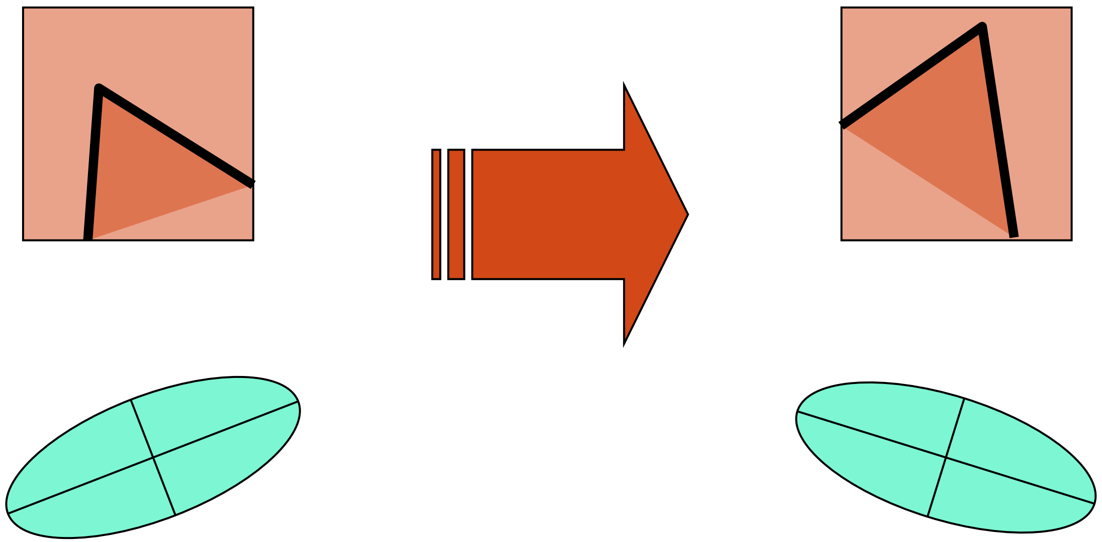
- Rotation



Ellipse rotates but its shape (i.e. eigenvalues) remains the same

HARRIS DETECTOR: INVARIANCE PROPERTIES

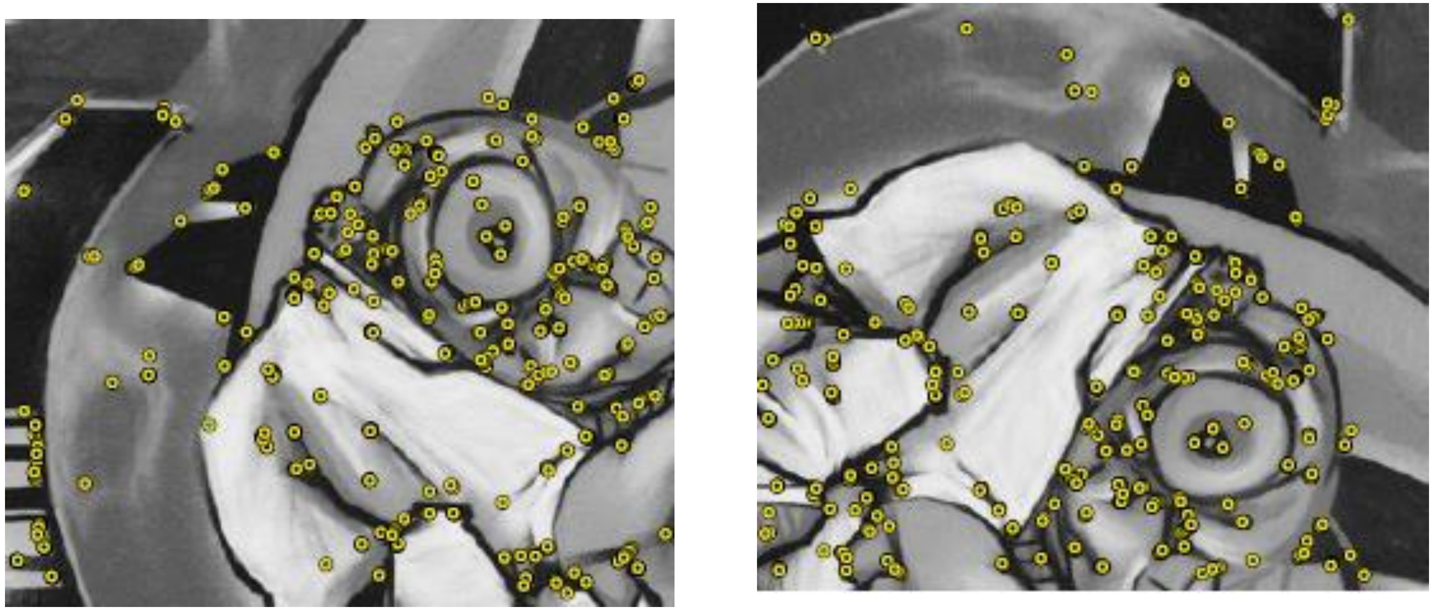
- Rotation



Ellipse rotates but its shape (i.e. eigenvalues) remains the same

Corner response is invariant to image rotation

HARRIS DETECTOR: INVARIANCE PROPERTIES

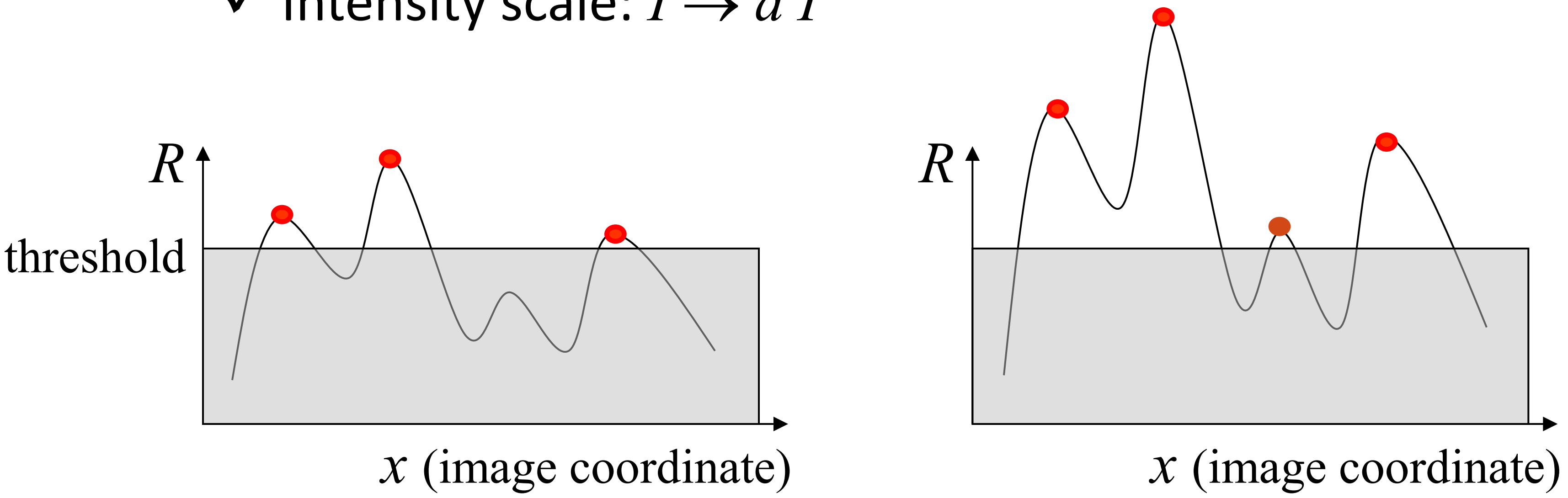


HARRIS DETECTOR: INVARIANCE PROPERTIES

- Affine intensity change: $I \rightarrow aI + b$
  - ✓ Only derivatives are used =>
    invariance to intensity shift $I \rightarrow I + b$

HARRIS DETECTOR: INVARIANCE PROPERTIES

- Affine intensity change: $I \rightarrow aI + b$
  - ✓ Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$
  - ✓ Intensity scale: $I \rightarrow a\,I$

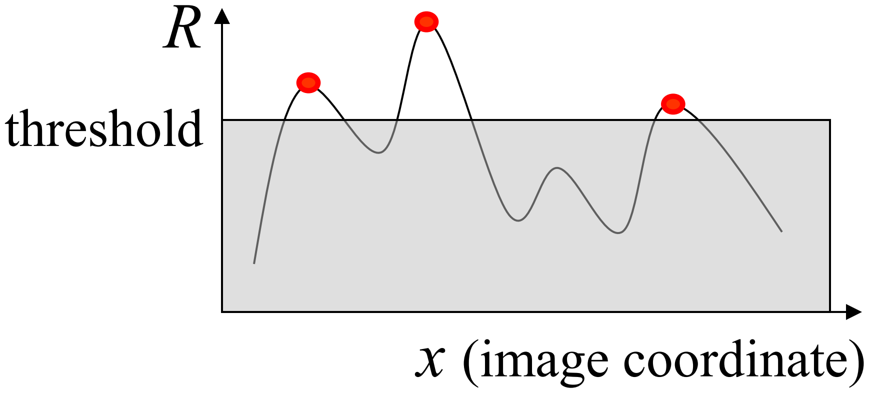


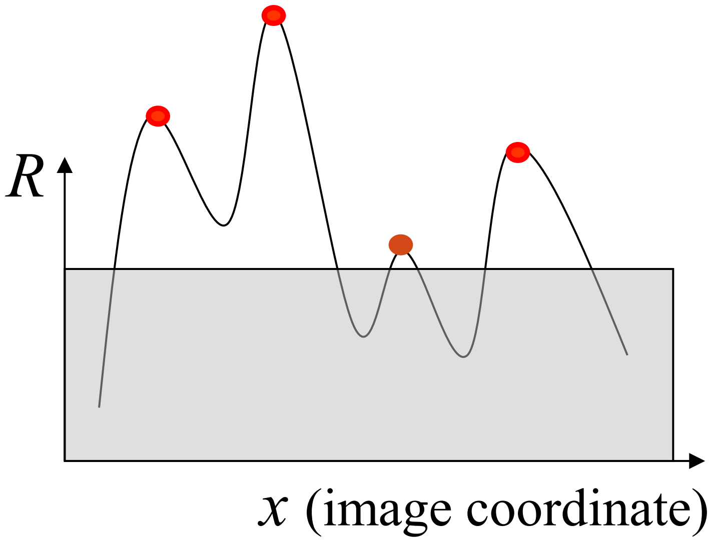
HARRIS DETECTOR: INVARIANCE PROPERTIES

- Affine intensity change: $I \to aI + b$

  ✓ Only derivatives are used =>
  invariance to intensity shift $I \to I + b$

  ✓ Intensity scale: $I \to a\,I$

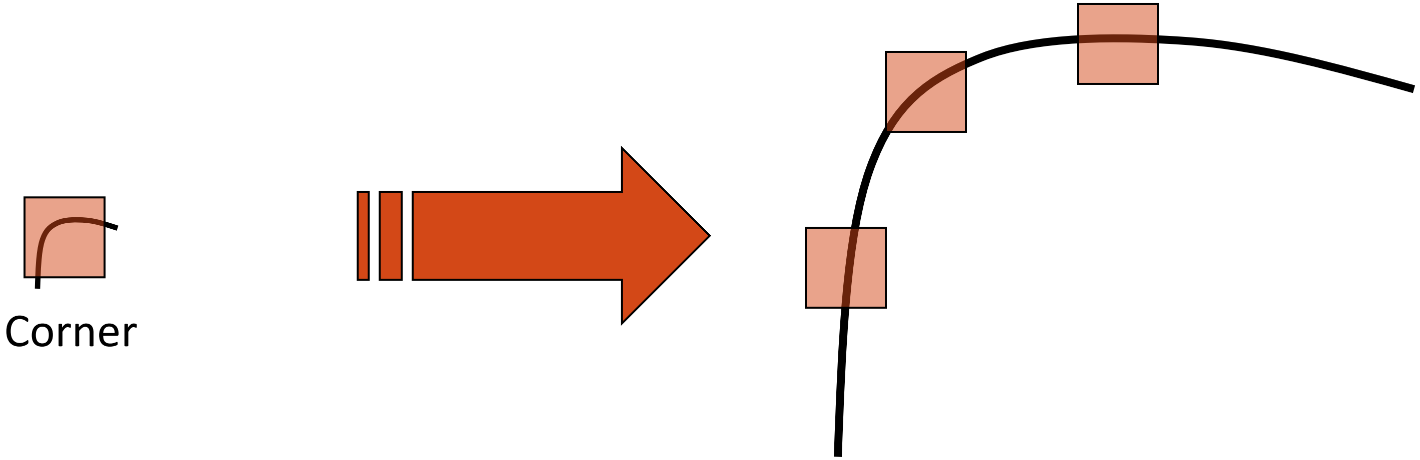




Partially invariant to affine intensity change

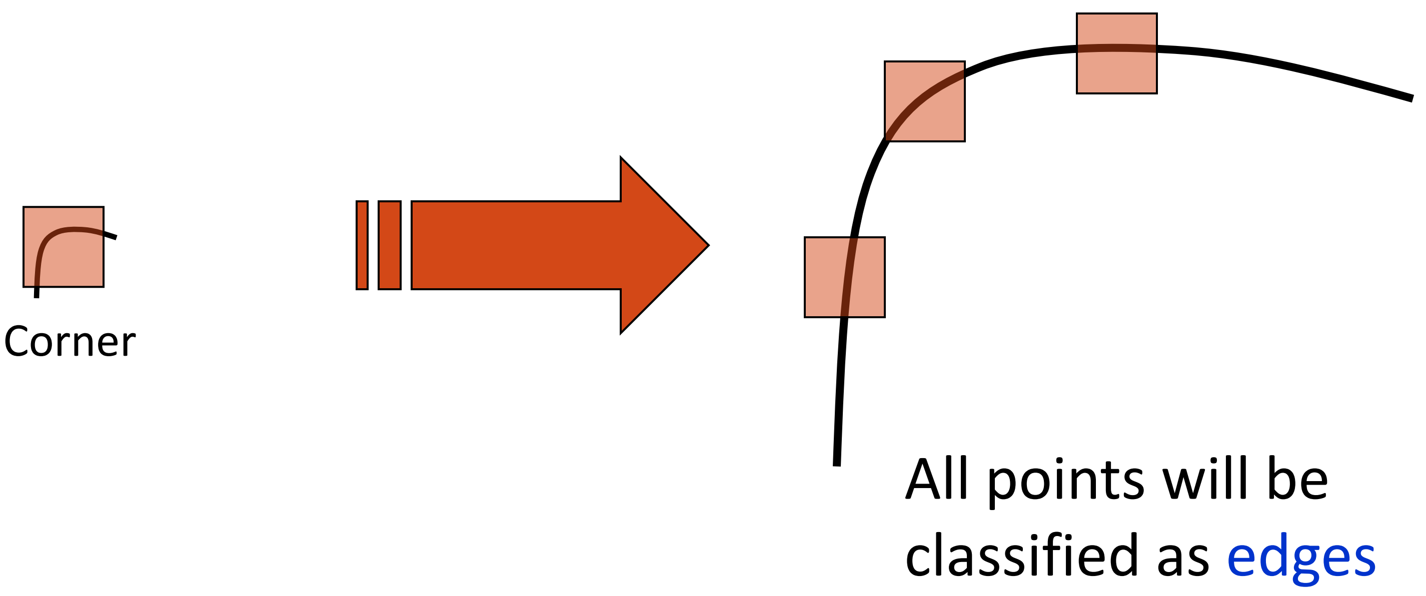
HARRIS DETECTOR: INVARIANCE PROPERTIES

- Scaling



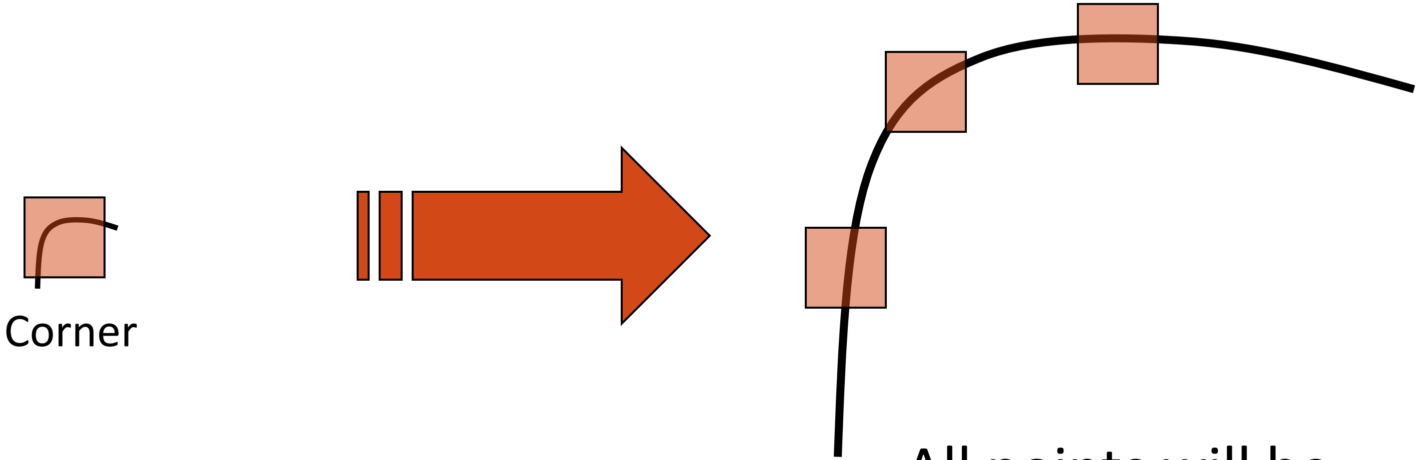
HARRIS DETECTOR: INVARIANCE PROPERTIES

- Scaling



HARRIS DETECTOR: INVARIANCE PROPERTIES

- Scaling

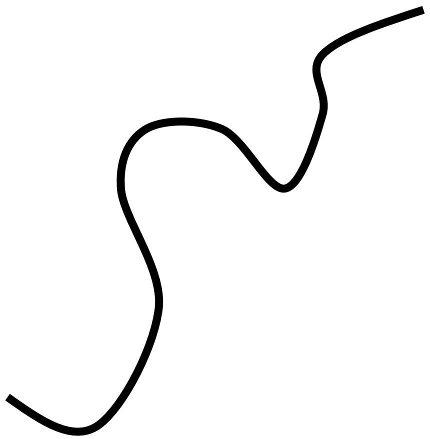


All points will be classified as edges

Not invariant to scaling

SCALE INVARIANT DETECTION

Suppose you're looking for corners

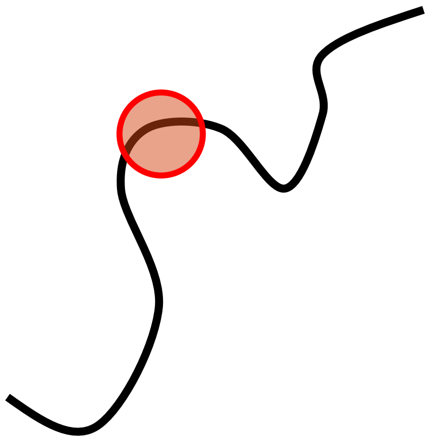


Key idea: find scale that gives local maximum of *f*

- in both position and scale
- One definition of *f*: the Harris operator

SCALE INVARIANT DETECTION

Suppose you're looking for corners

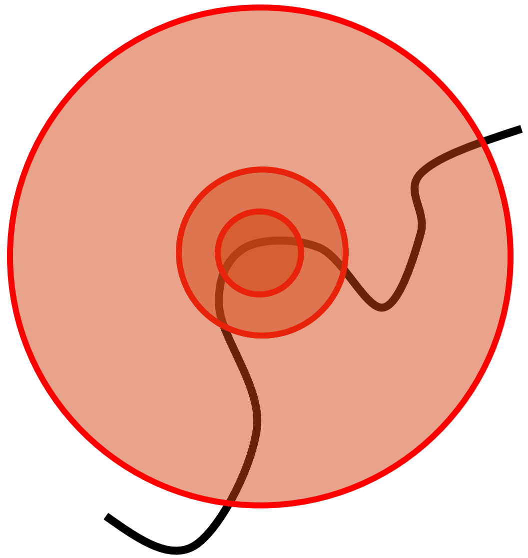


Key idea: find scale that gives local maximum of *f*

- in both position and scale
- One definition of *f*: the Harris operator

SCALE INVARIANT DETECTION

Suppose you're looking for corners



Key idea: find scale that gives local maximum of *f*

- in both position and scale
- One definition of *f*: the Harris operator

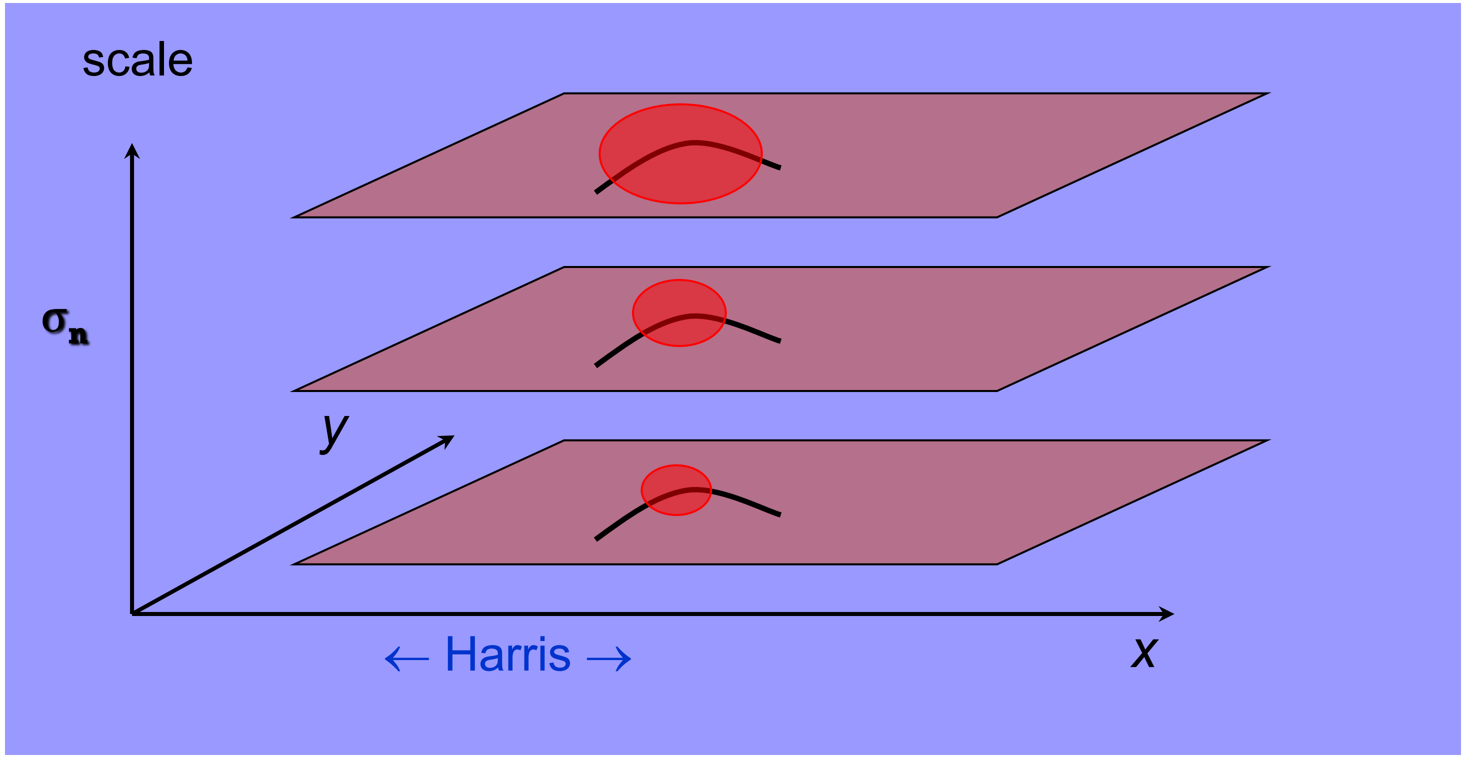
MULTI-SCALE HARRIS DETECTOR

- Detects interest points at varying scales.

$$R(H_W) = \det(H_W(x,y,\sigma_I,\sigma_D)) - \alpha\ \mathrm{trace}^2(H_W(x,y,\sigma_I,\sigma_D))$$

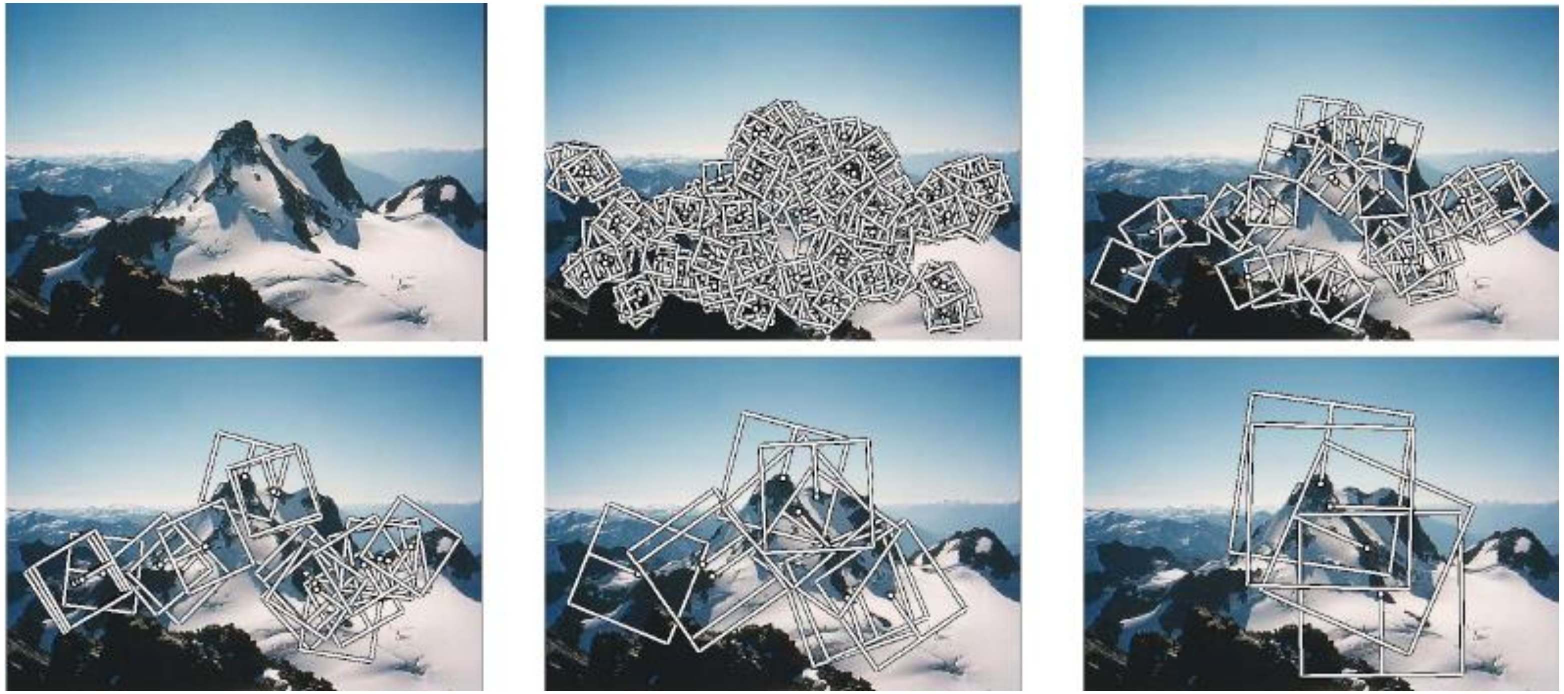
$\sigma_n = k^n \sigma$

$\sigma_D = \sigma_n$
$\sigma_I = \gamma \sigma_D$



MULTI-SCALE HARRIS DETECTOR (CONT'D)

Interest points detected at varying scales:

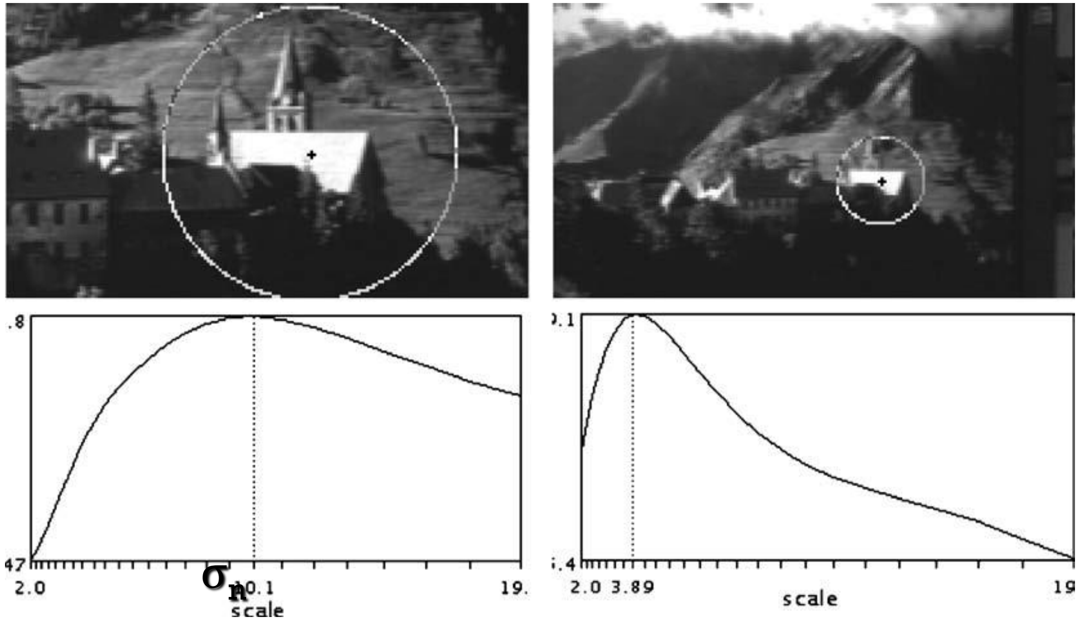


M. Brown, R. Szeliski, and S. Winder, "Multi-image matching using multi-scale oriented Patches", *IEEE Conference on Computer Vision and Pattern Recognition,* vol. I, pages 510-517, 2005.

AUTOMATIC SCALE SELECTION

- Design a function $\mathbf{F(x,\sigma_n)}$ which provides some local measure.
- Select points at which $\mathbf{F(x,\sigma_n)}$ is maximal over σ_n.

$\mathbf{F(x,\sigma_n)}$

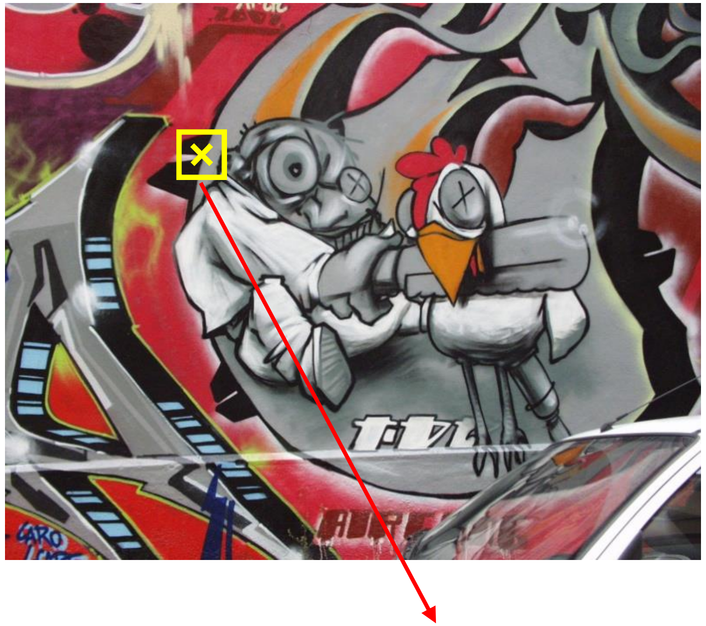


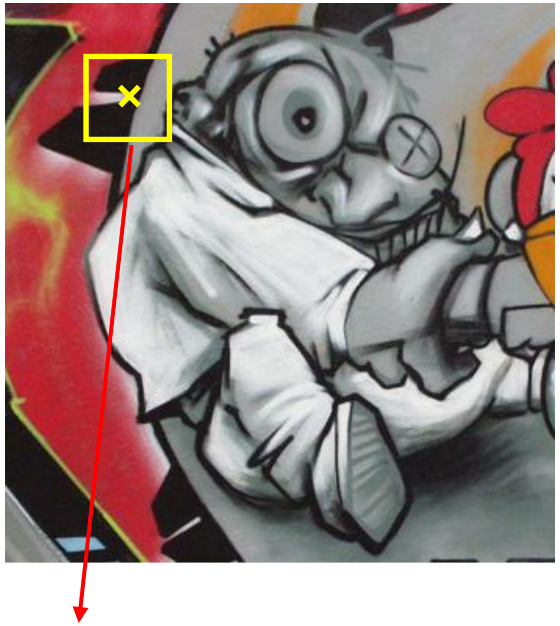
σ_n

max of $\mathbf{F(x,\sigma_n)}$ corresponds to characteristic scale!

T. Lindeberg, "Feature detection with automatic scale selection" *International Journal of Computer Vision,* vol. 30, no. 2, pp 77-116, 1998.

AUTOMATIC SCALE SELECTION





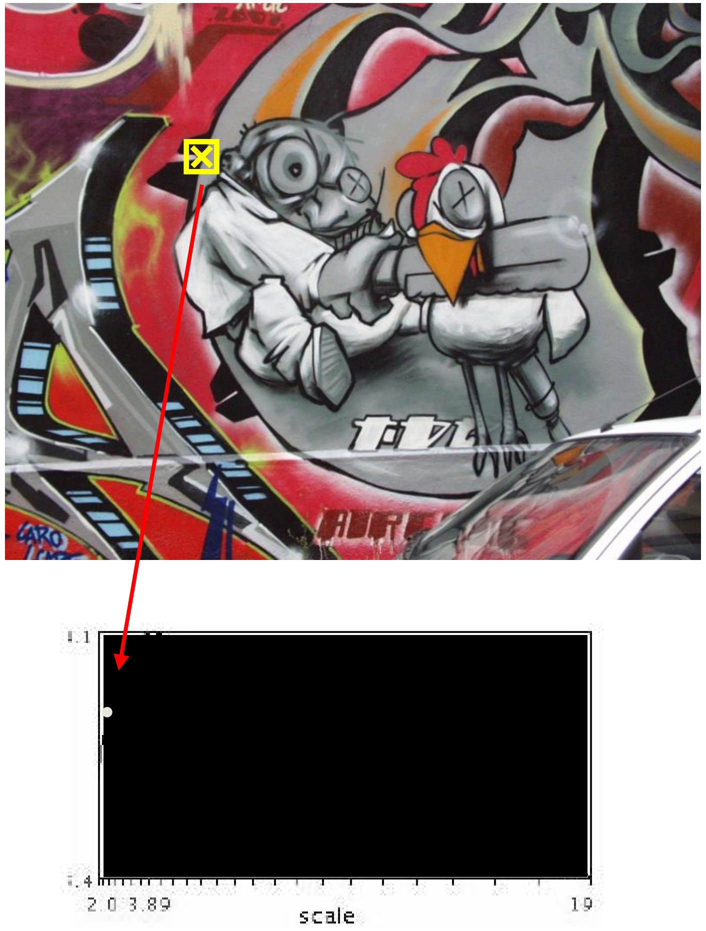
$$f(I_{i_1 \ldots i_m}(x,\sigma)) = f(I_{i_1 \ldots i_m}(x',\sigma'))$$

How to find corresponding patch sizes?

K. Grauman, B. Leibe

AUTOMATIC SCALE SELECTION

- Function responses for increasing scale (scale signature)



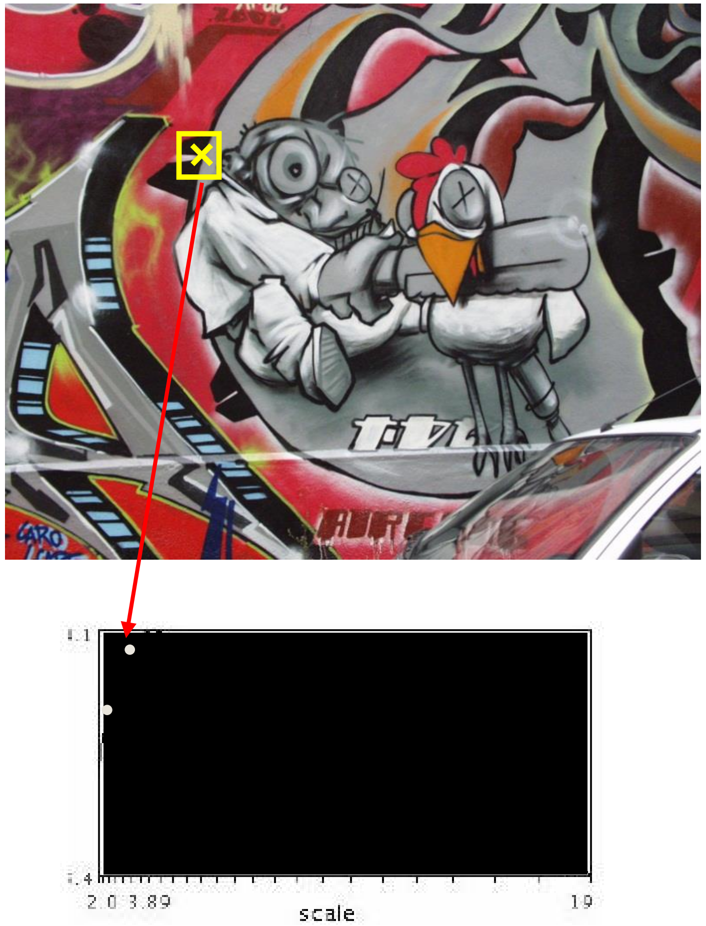
$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

$$f(I_{i_1 \ldots i_m}(x',\sigma))$$

K. Grauman, B. Leibe

AUTOMATIC SCALE SELECTION

- Function responses for increasing scale (scale signature)



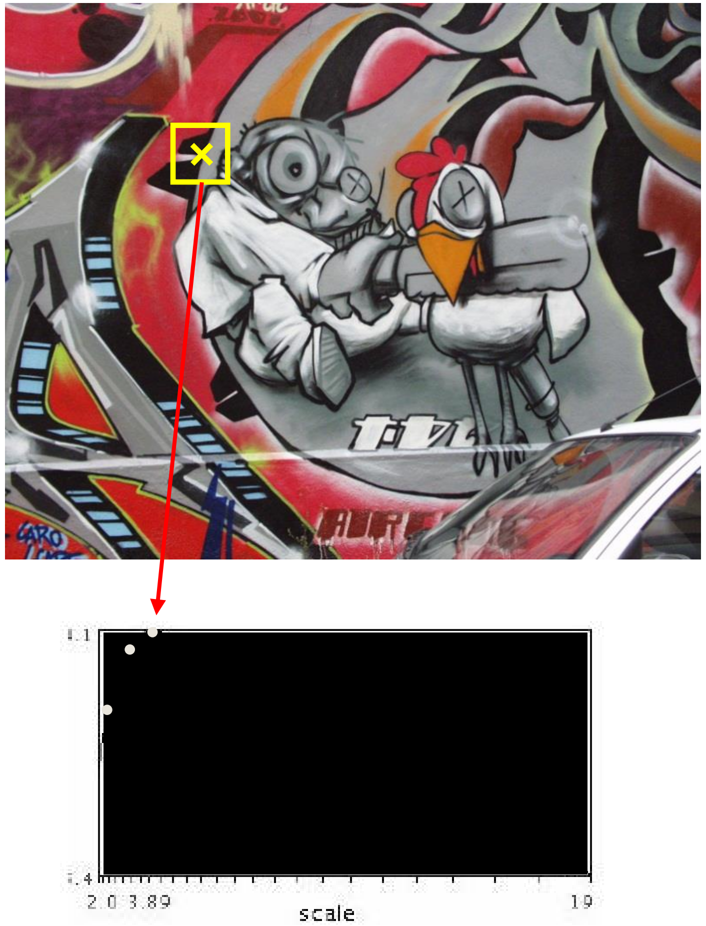
$f(I_{i_1 \ldots i_m}(x, \sigma))$

$f(I_{i_1 \ldots i_m}(x', \sigma))$

K. Grauman, B. Leibe

AUTOMATIC SCALE SELECTION

- Function responses for increasing scale (scale signature)



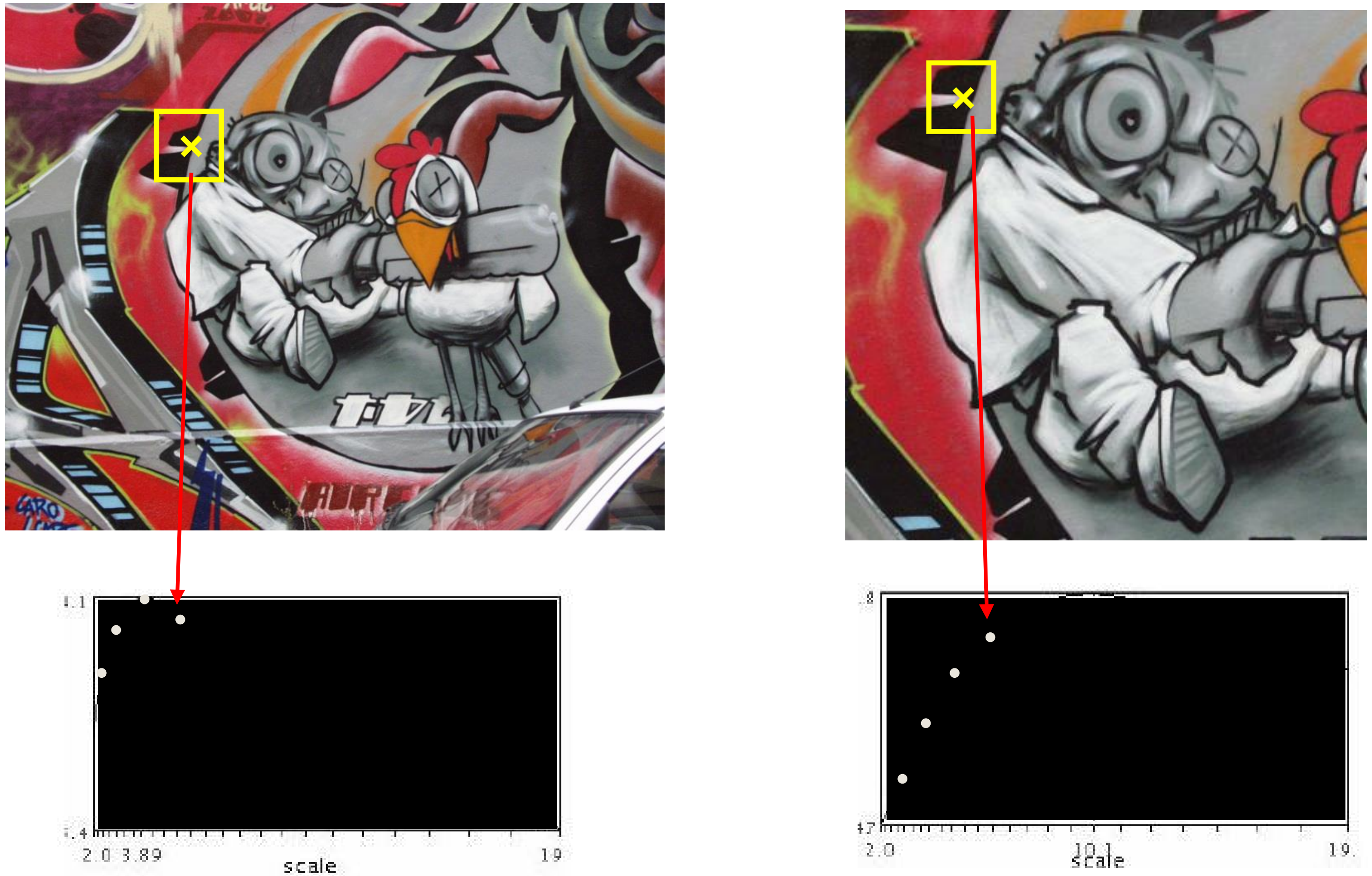
$f(I_{i_1 \dots i_m}(x,\sigma))$

$f(I_{i_1 \dots i_m}(x',\sigma))$

K. Grauman, B. Leibe

AUTOMATIC SCALE SELECTION

- Function responses for increasing scale (scale signature)

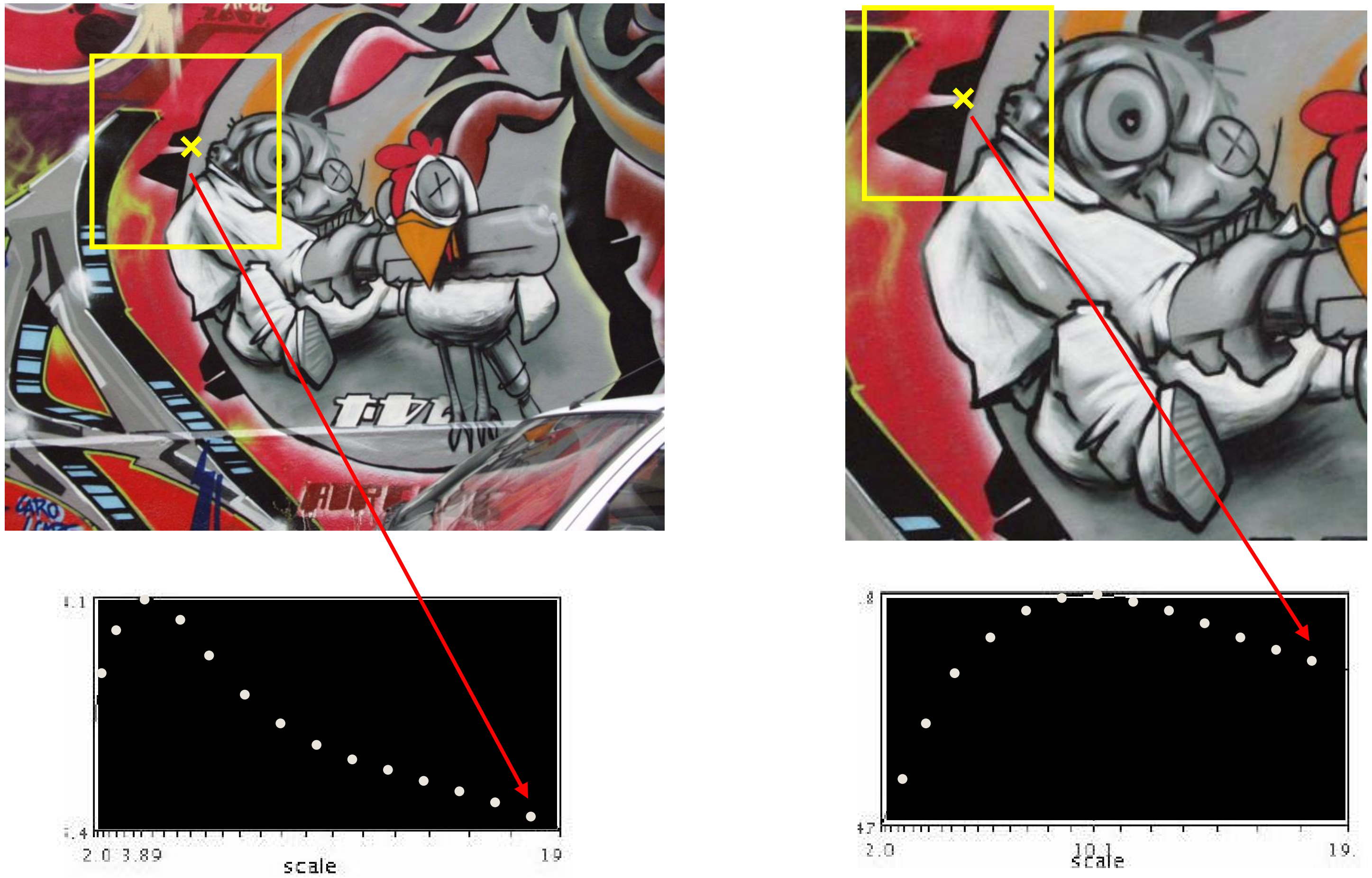


$f(I_{i_1 \ldots i_m}(x, \sigma))$ $f(I_{i_1 \ldots i_m}(x', \sigma))$

K. Grauman, B. Leibe

AUTOMATIC SCALE SELECTION

- Function responses for increasing scale (scale signature)



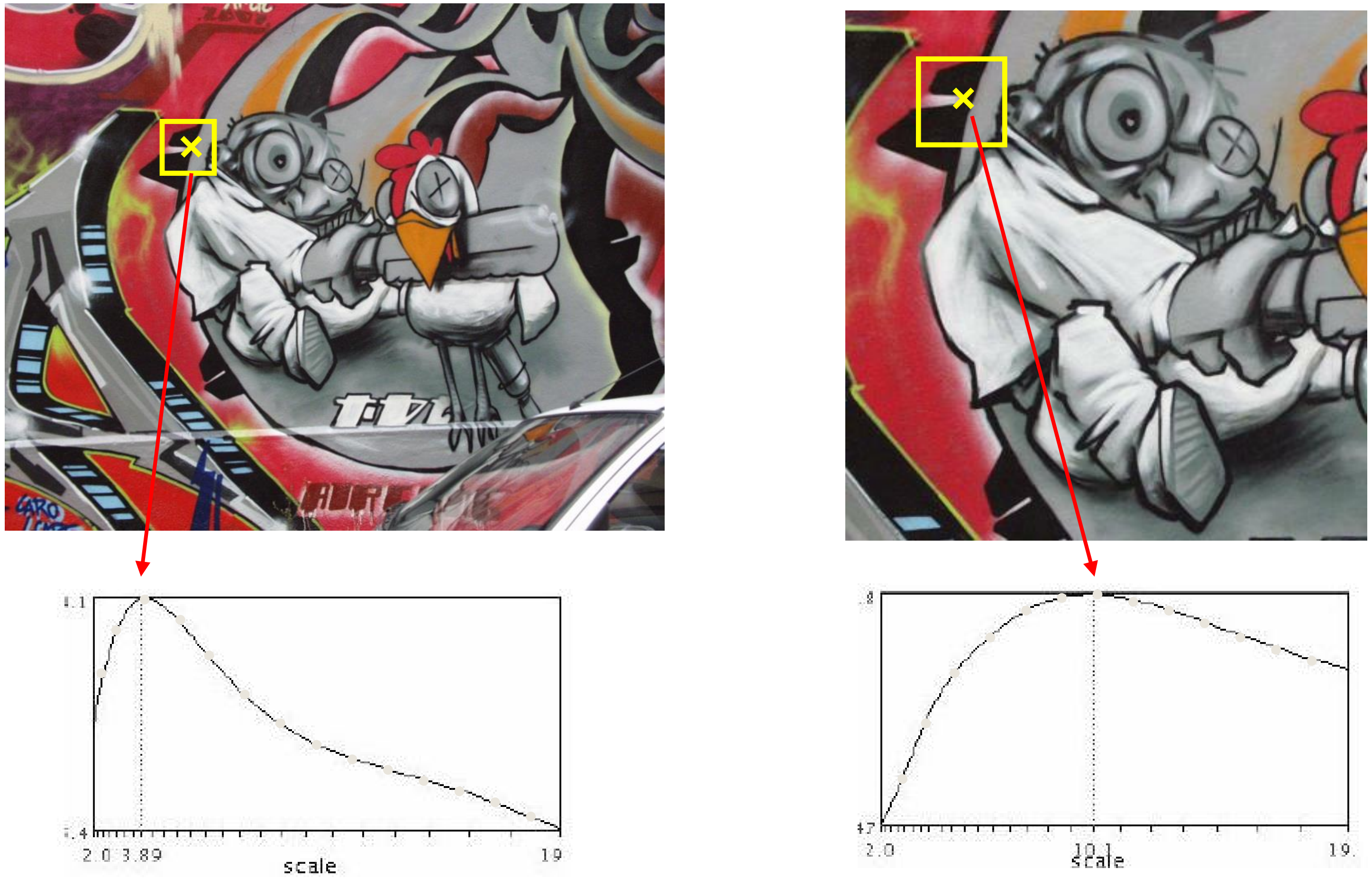
$f(I_{i_1 \ldots i_m}(x, \sigma))$

$f(I_{i_1 \ldots i_m}(x', \sigma))$

K. Grauman, B. Leibe

AUTOMATIC SCALE SELECTION

- Function responses for increasing scale (scale signature)



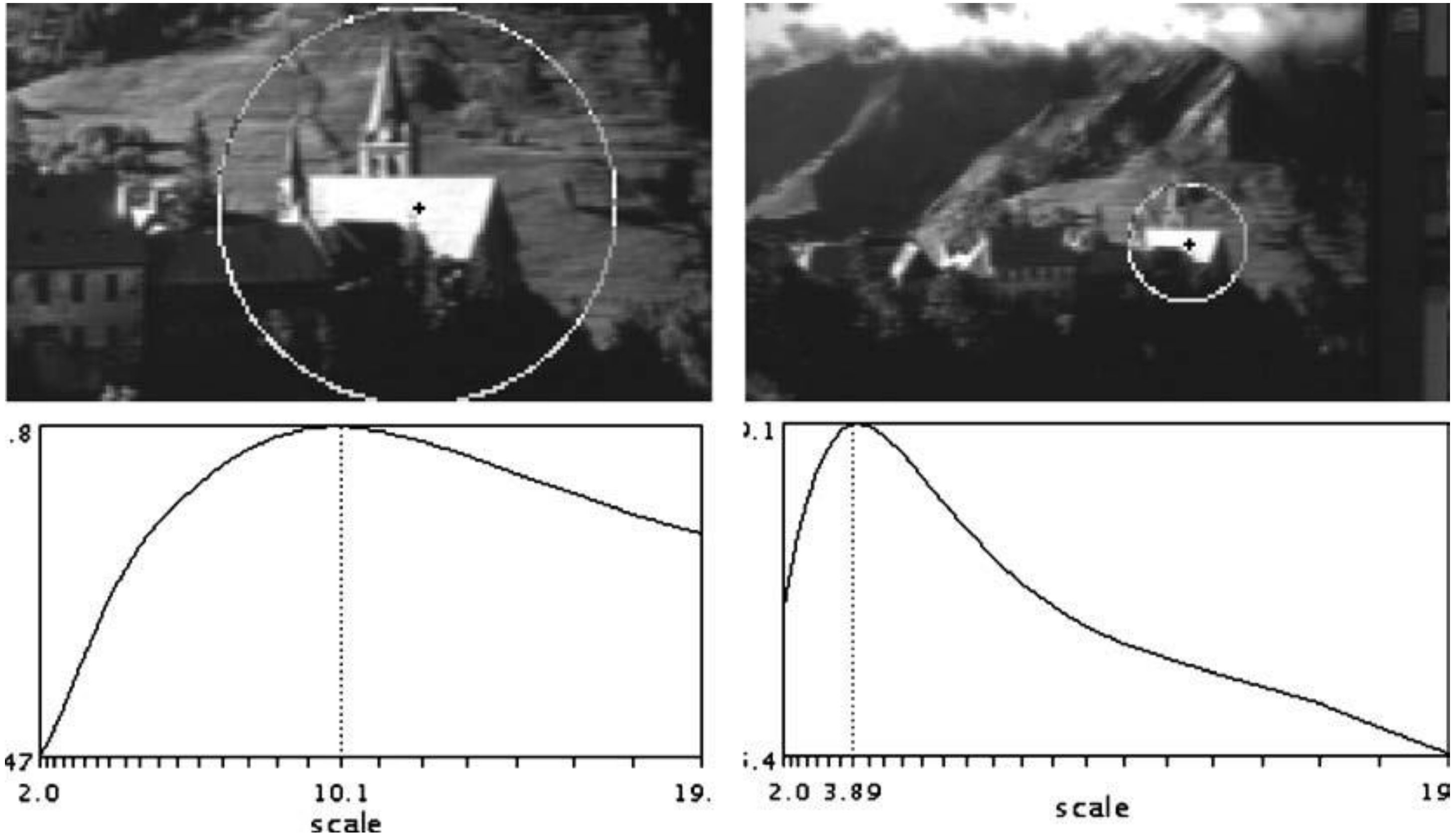
$f(I_{i_1 \ldots i_m}(x,\sigma))$ $f(I_{i_1 \ldots i_m}(x',\sigma'))$

K. Grauman, B. Leibe

AUTOMATIC SCALE SELECTION

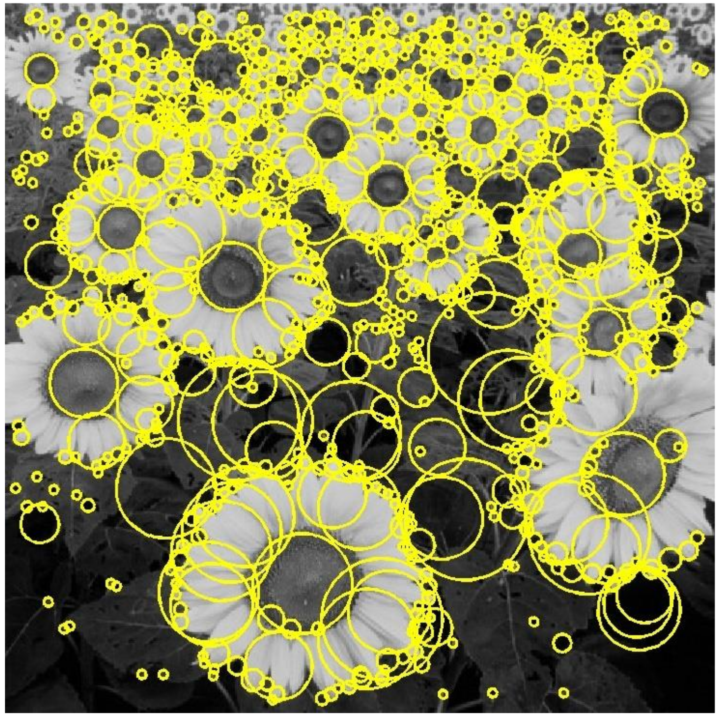
KEYPOINT DETECTION WITH SCALE SELECTION

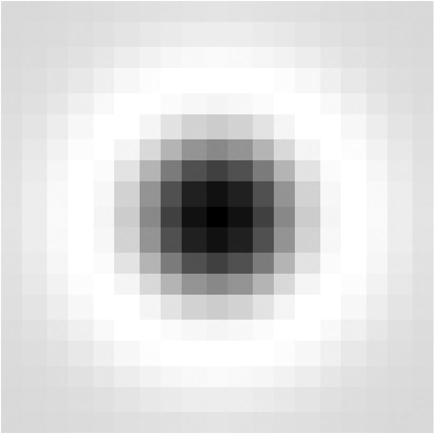
We want to extract keypoints with characteristic scale that is *covariant* with the image transformation



BASIC IDEA

Convolve the image with a "blob filter" at multiple scales and look for extrema of filter response in the resulting *scale space*

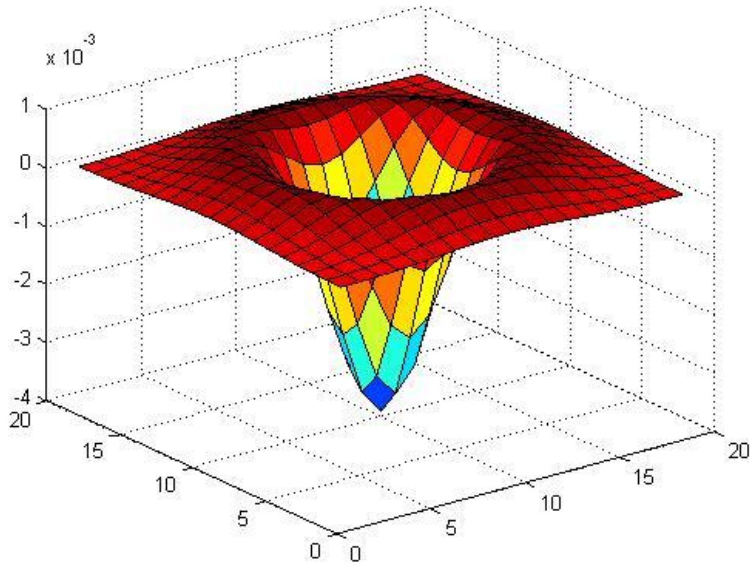


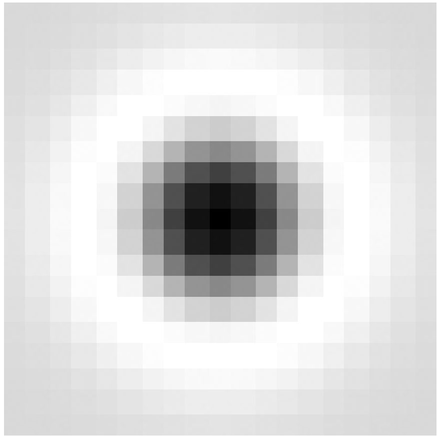


T. Lindeberg. Feature detection with automatic scale selection. *IJCV* 30(2), pp 77-116, 1998.

BLOB FILTER

- **Laplacian of Gaussian:** Circularly symmetric operator for blob detection in 2D



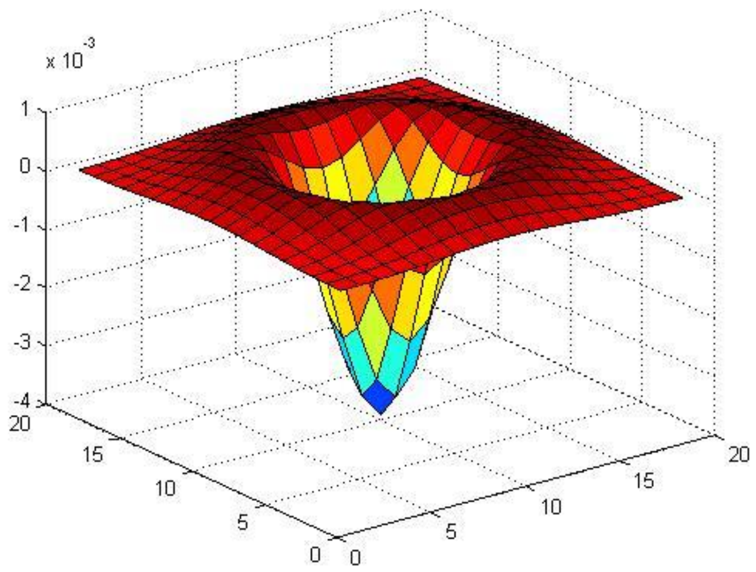


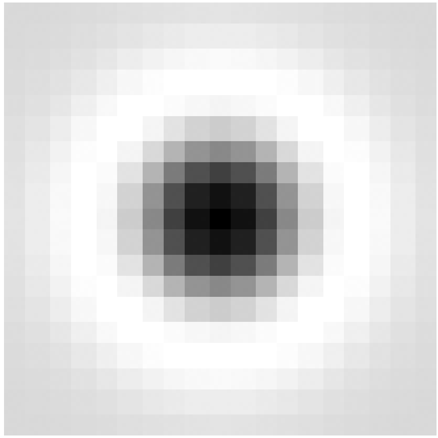
$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

BLOB DETECTION IN 2D

- *Scale-normalized* Laplacian of Gaussian:

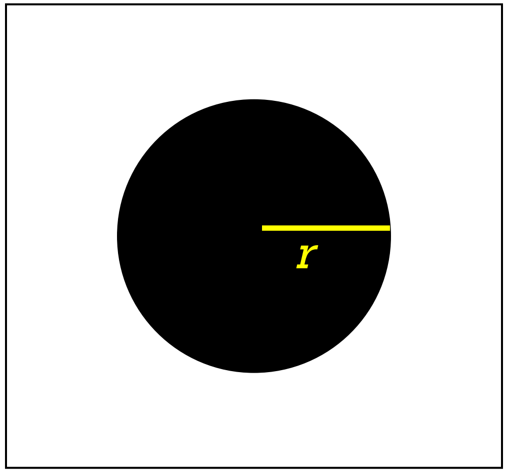
$$\nabla^2_{\text{norm}} g = \sigma^2 \left(\frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \right)$$



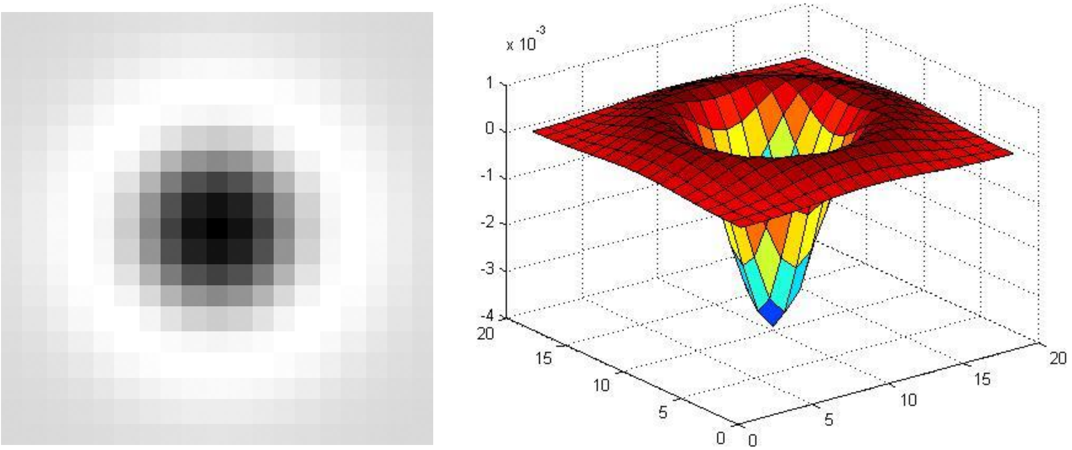


BLOB DETECTION IN 2D

- At what scale does the Laplacian achieve a maximum response to a binary circle of radius r?



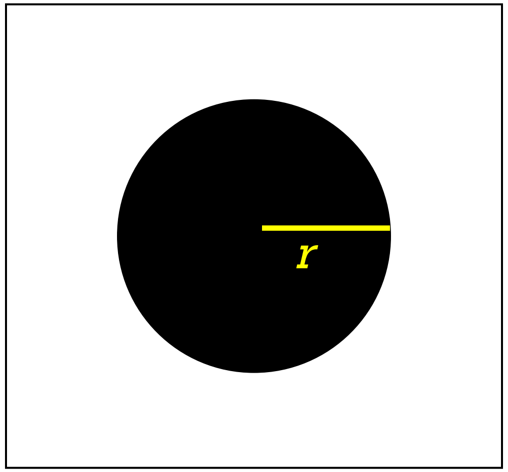
image



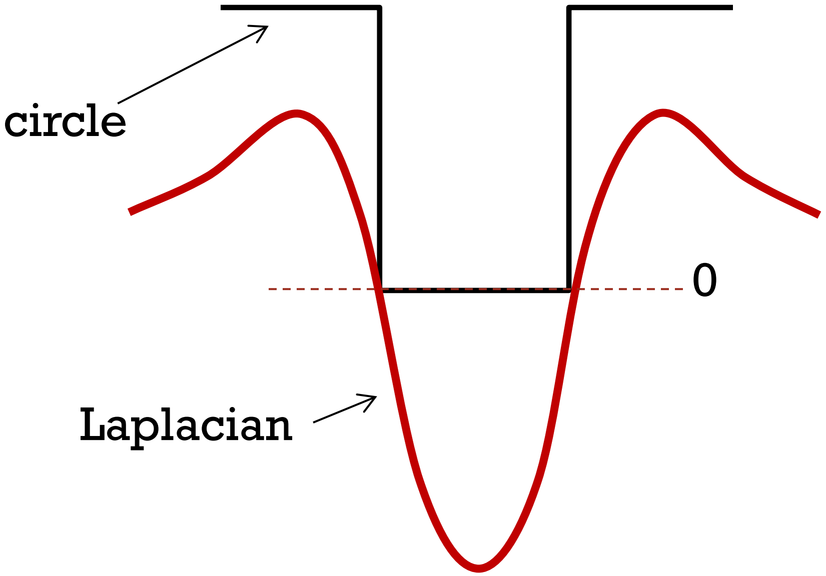
Laplacian

BLOB DETECTION IN 2D

- At what scale does the Laplacian achieve a maximum response to a binary circle of radius r?
- To get maximum response, the zeros of the Laplacian have to be aligned with the circle



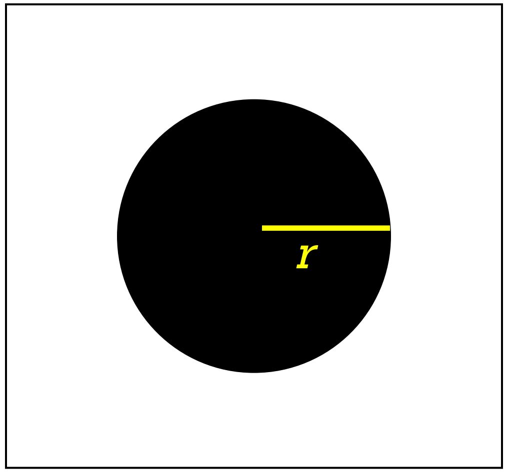
image



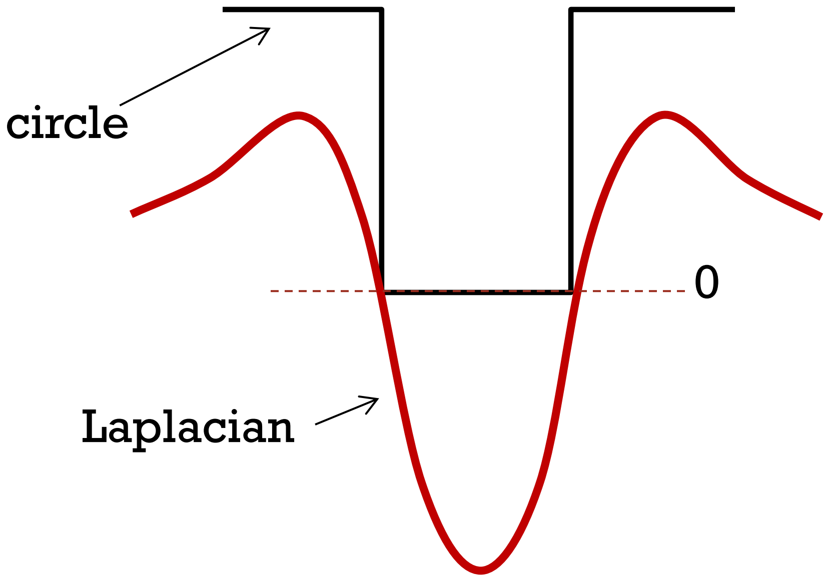
BLOB DETECTION IN 2D

- At what scale does the Laplacian achieve a maximum response to a binary circle of radius r?
- To get maximum response, the zeros of the Laplacian have to be aligned with the circle
- The Laplacian is given by (up to scale):

$$(x^2 + y^2 - 2\sigma^2)\, e^{-(x^2+y^2)/2\sigma^2}$$



image

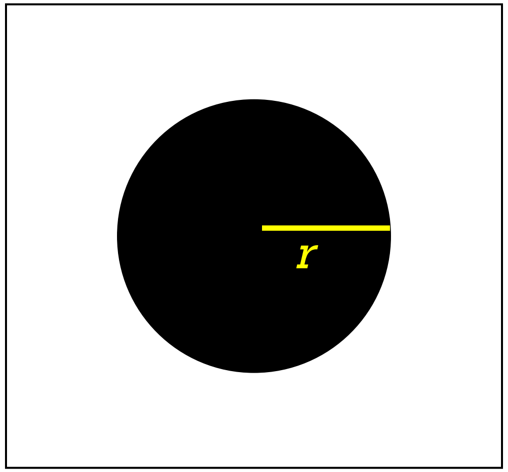


BLOB DETECTION IN 2D

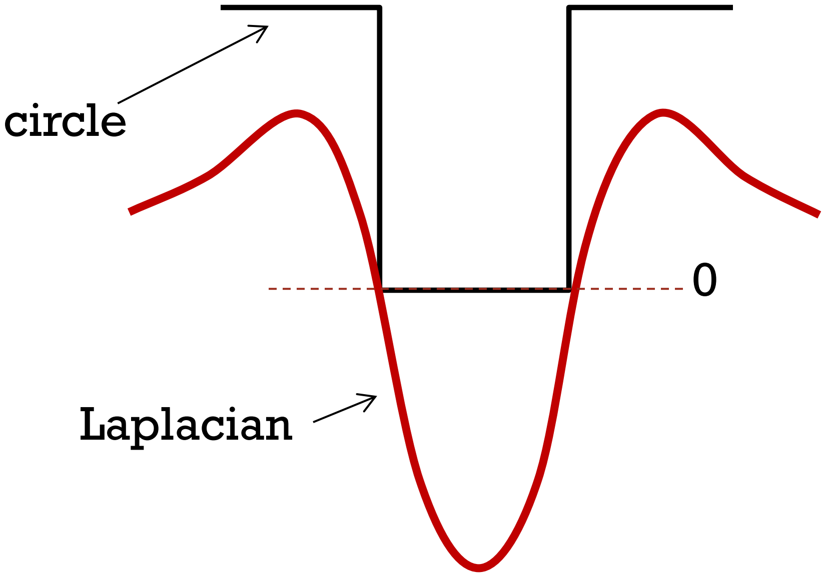
- At what scale does the Laplacian achieve a maximum response to a binary circle of radius r?
- To get maximum response, the zeros of the Laplacian have to be aligned with the circle
- The Laplacian is given by (up to scale):

$$(x^2 + y^2 - 2\sigma^2)\, e^{-(x^2+y^2)/2\sigma^2}$$

- Therefore, the maximum response occurs at $\sigma = r / \sqrt{2}$.

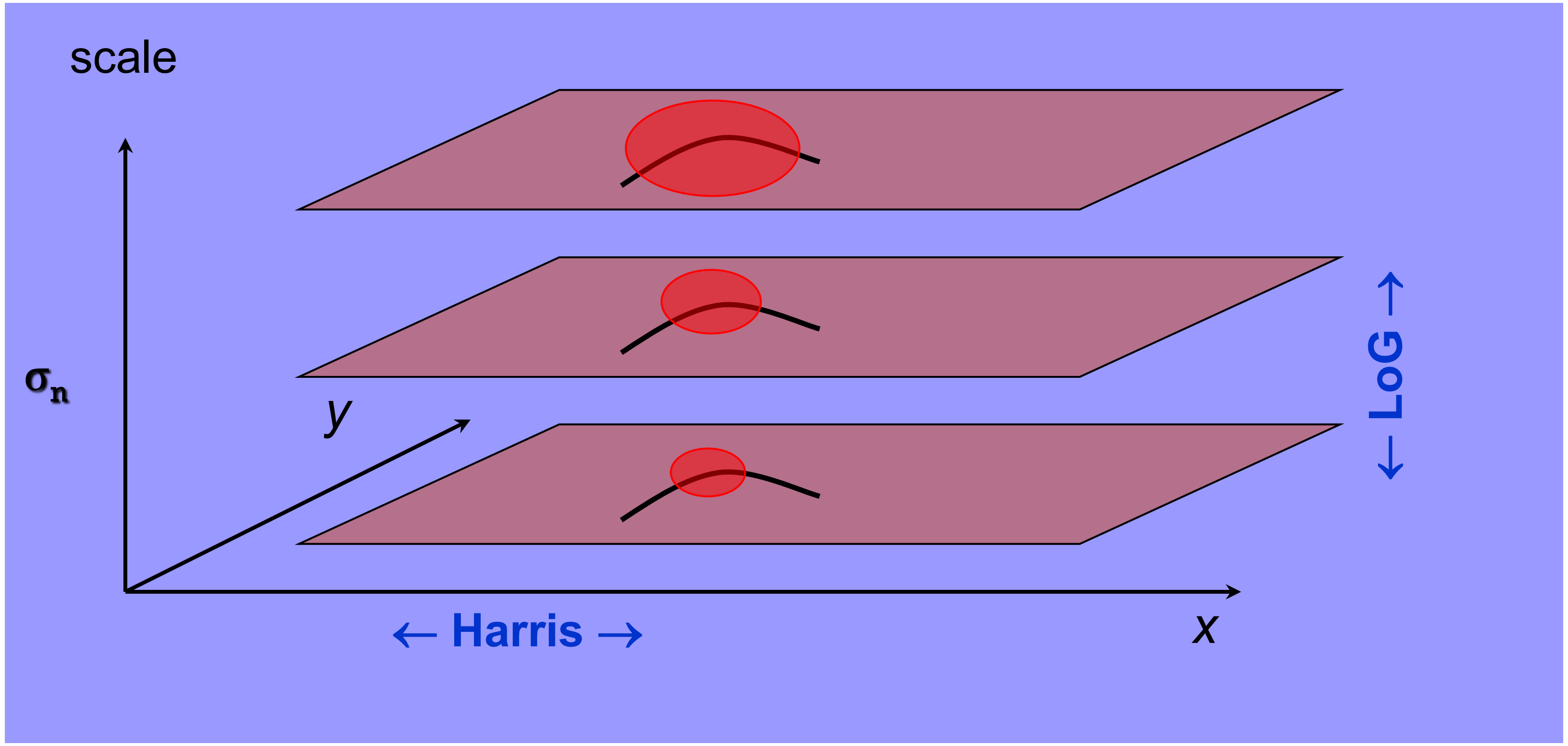


image



HARRIS-LAPLACE DETECTOR

- Multi-scale Harris with scale selection.
- Uses LoG maxima to find characteristic scale.

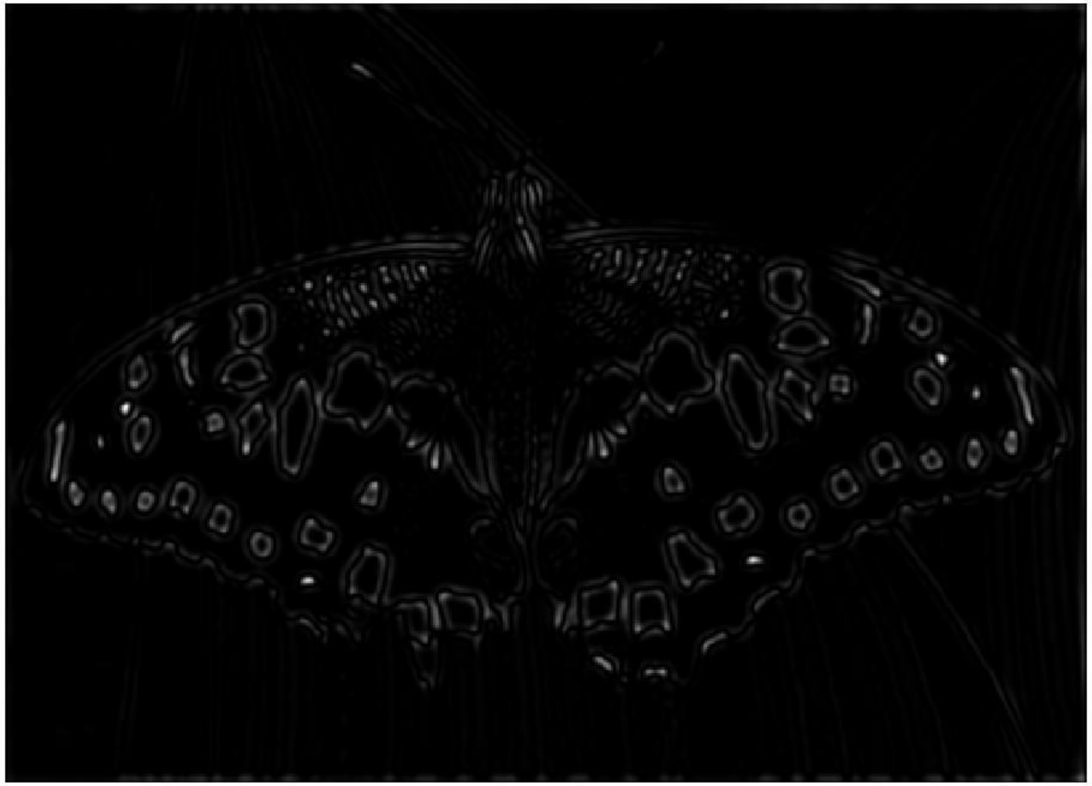


SCALE-SPACE BLOB DETECTOR

1. Convolve image with scale-normalized Laplacian at several scales

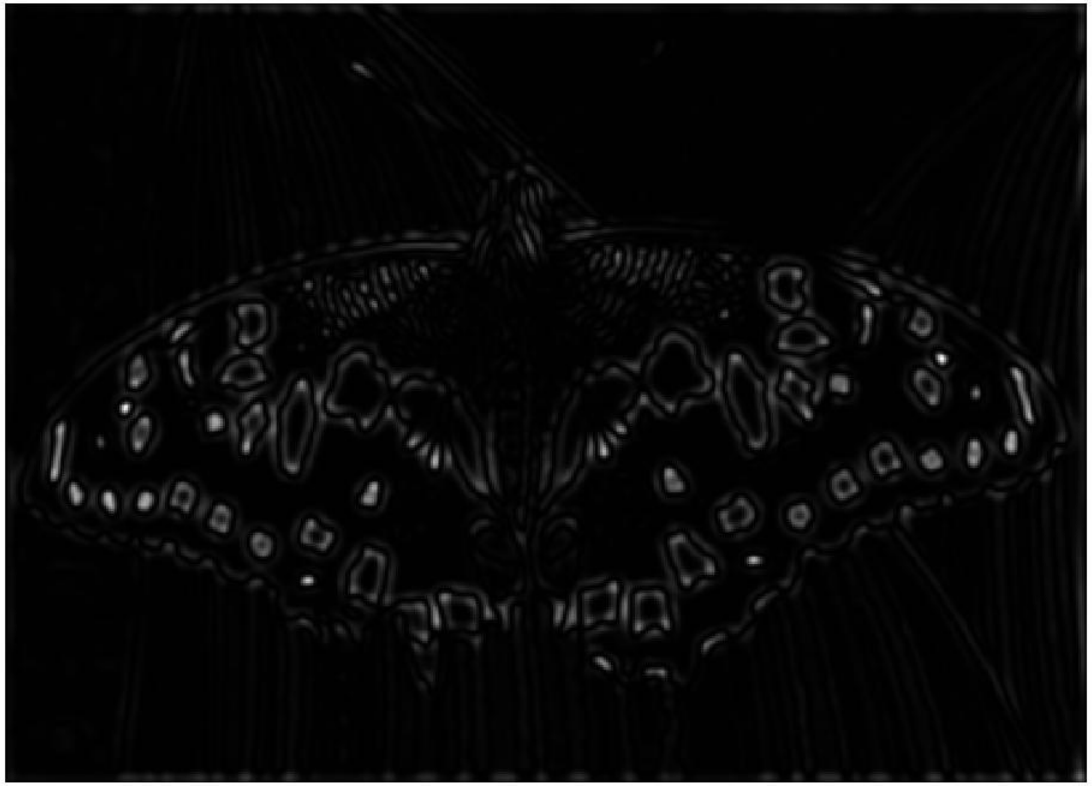
SCALE-SPACE BLOB DETECTOR: EXAMPLE

SCALE-SPACE BLOB DETECTOR: EXAMPLE



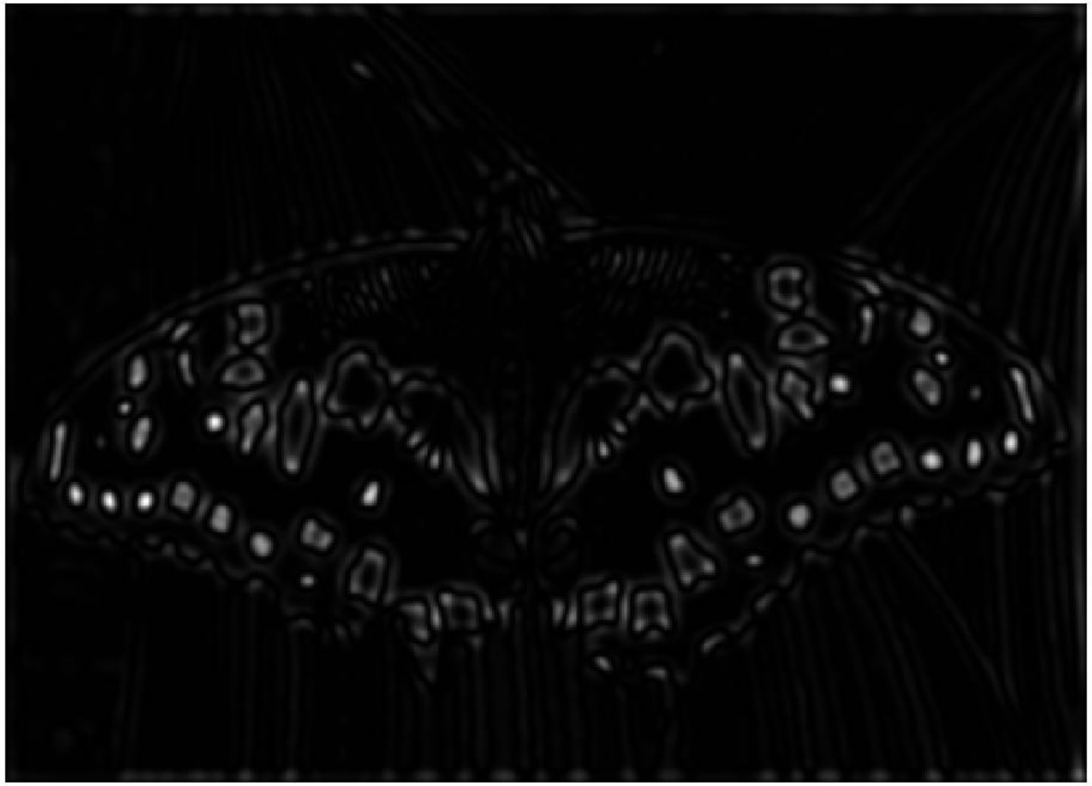
sigma = 2

SCALE-SPACE BLOB DETECTOR: EXAMPLE



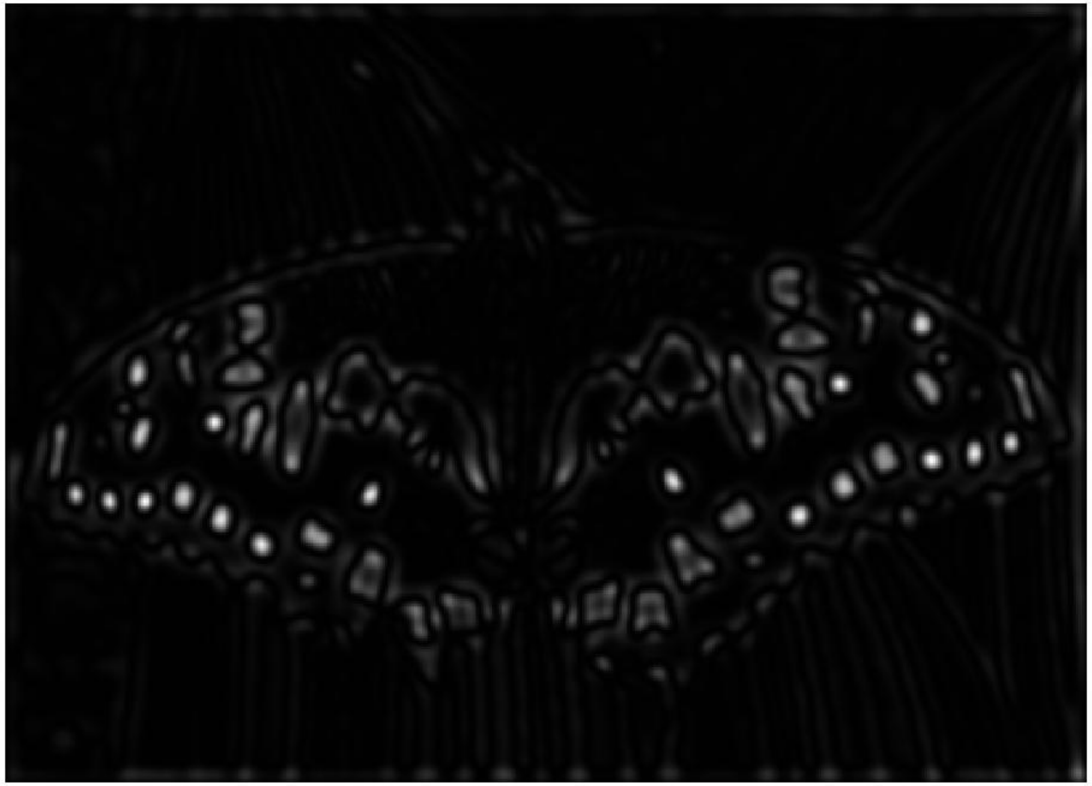
sigma = 2.5018

SCALE-SPACE BLOB DETECTOR: EXAMPLE



sigma = 3.1296

SCALE-SPACE BLOB DETECTOR: EXAMPLE

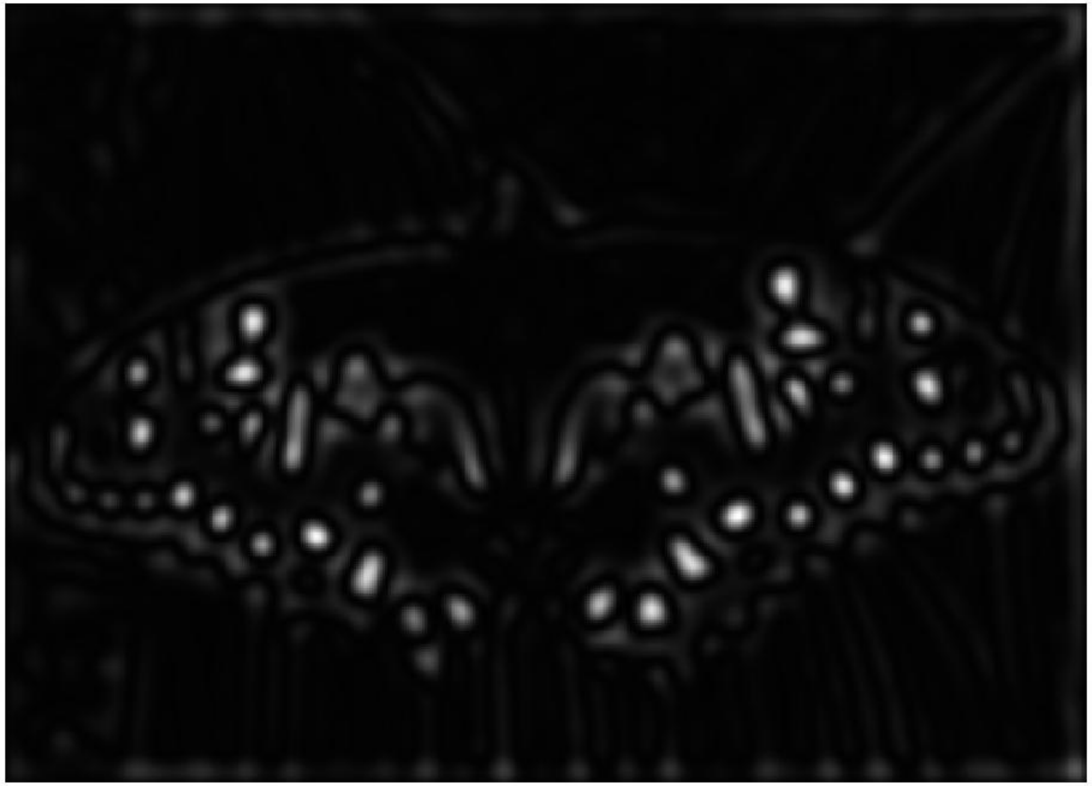


sigma = 3.9149

SCALE-SPACE BLOB DETECTOR: EXAMPLE

sigma = 4.8972

SCALE-SPACE BLOB DETECTOR: EXAMPLE



sigma = 6.126

SCALE-SPACE BLOB DETECTOR: EXAMPLE

sigma = 7.6631

SCALE-SPACE BLOB DETECTOR: EXAMPLE

sigma = 9.5859

SCALE-SPACE BLOB DETECTOR: EXAMPLE

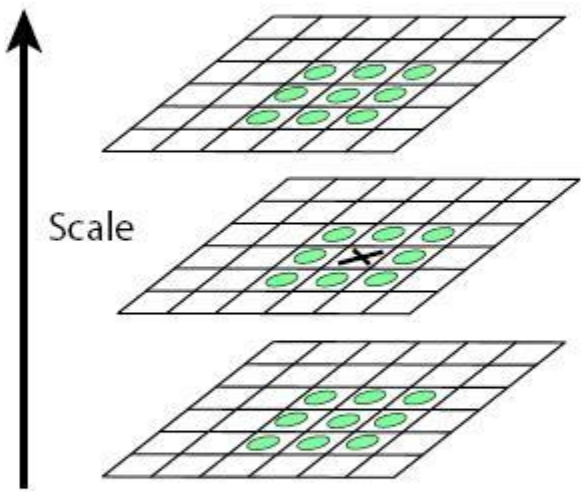
sigma = 11.9912

SCALE-SPACE BLOB DETECTOR: EXAMPLE

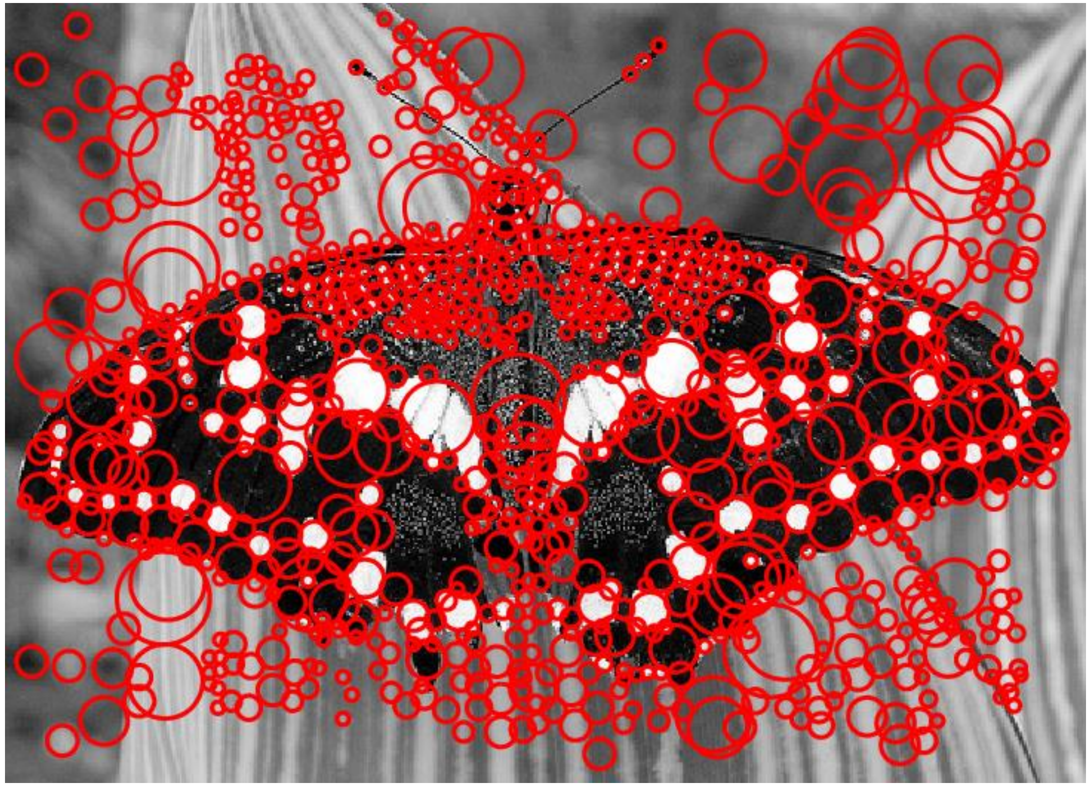
sigma = 15

SCALE-SPACE BLOB DETECTOR

1. Convolve image with scale-normalized Laplacian at several scales
2. Find maxima of squared Laplacian response in scale-space



SCALE-SPACE BLOB DETECTOR: EXAMPLE

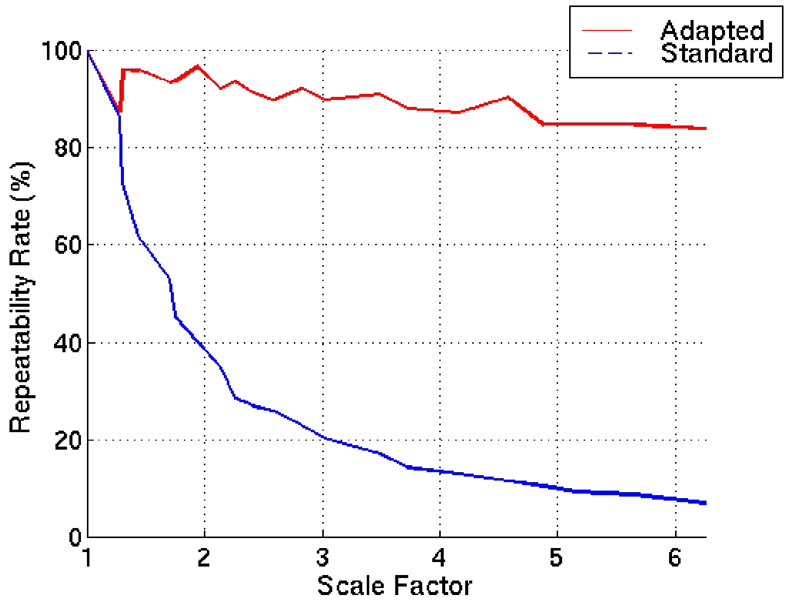


Using Laplacian of Gaussian (LoG)

HARRIS-LAPLACE DETECTOR (CONT'D)

- Invariant to:
  - Scale
  - Rotation
  - Translation
- Robust to:
  - Illumination changes
  - Limited viewpoint changes

Repeatability



EFFICIENT IMPLEMENTATION (SIFT)

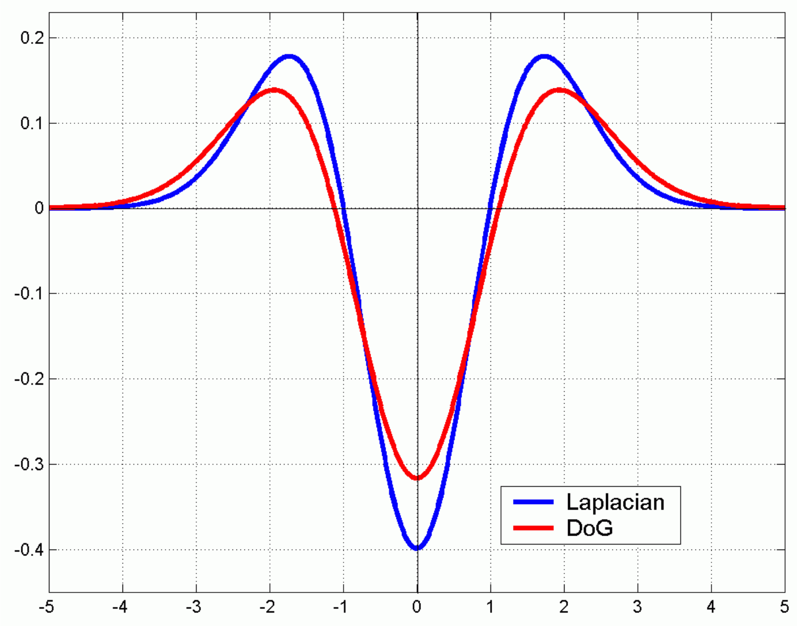
- Approximating the Laplacian with a difference of Gaussians by SIFT detector:

$$L = \sigma^2 \left(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right)$$

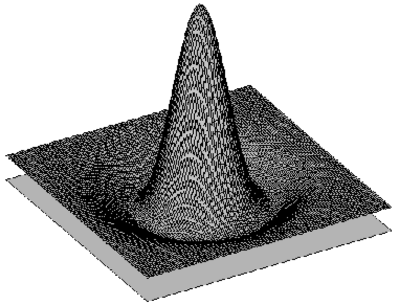
(Laplacian)

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)



DIFFERENCE-OF-GAUSSIAN (DOG)



K. Grauman, B. Leibe

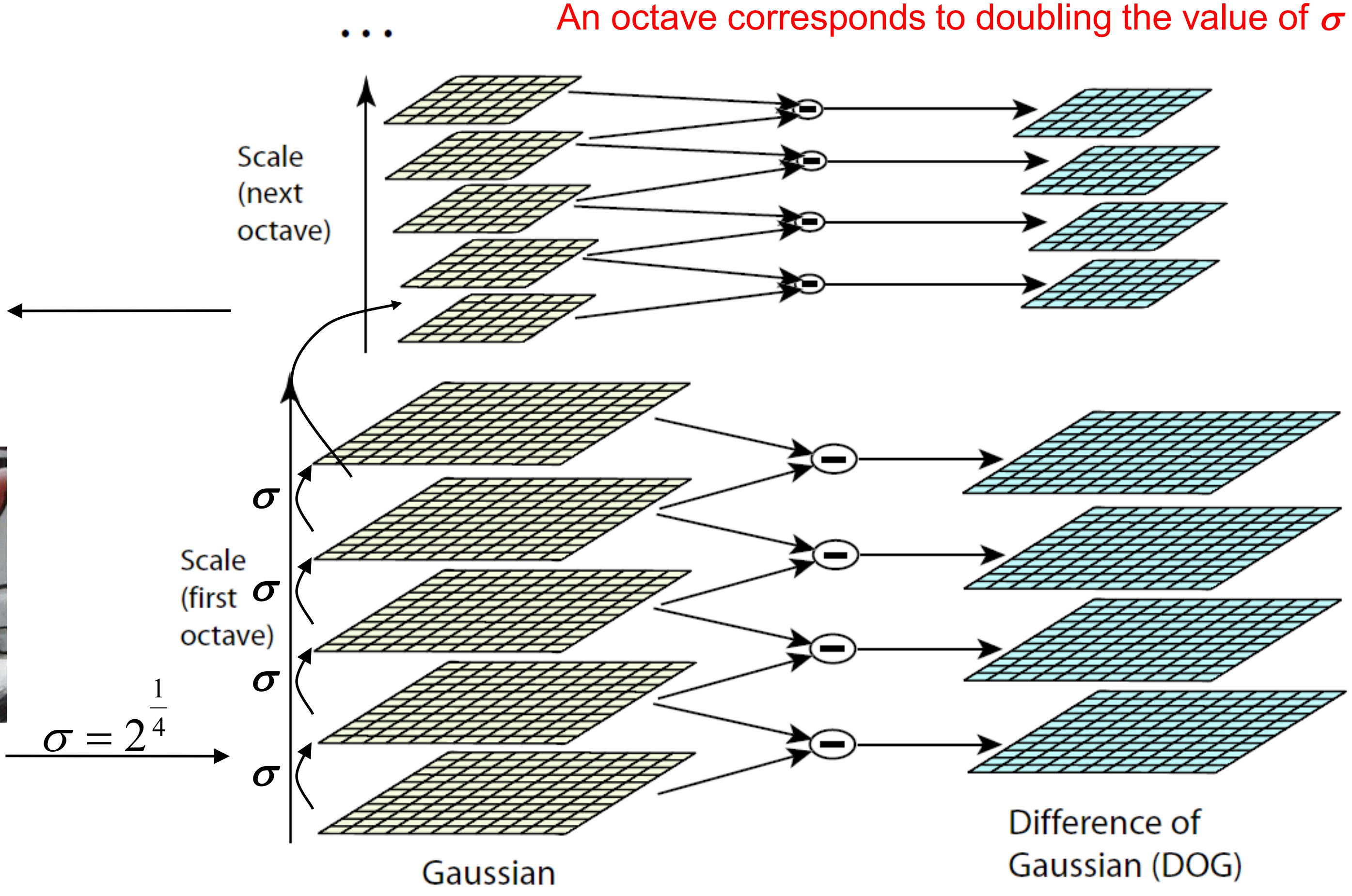
DOG – EFFICIENT COMPUTATION

- Computation in Gaussian scale pyramid

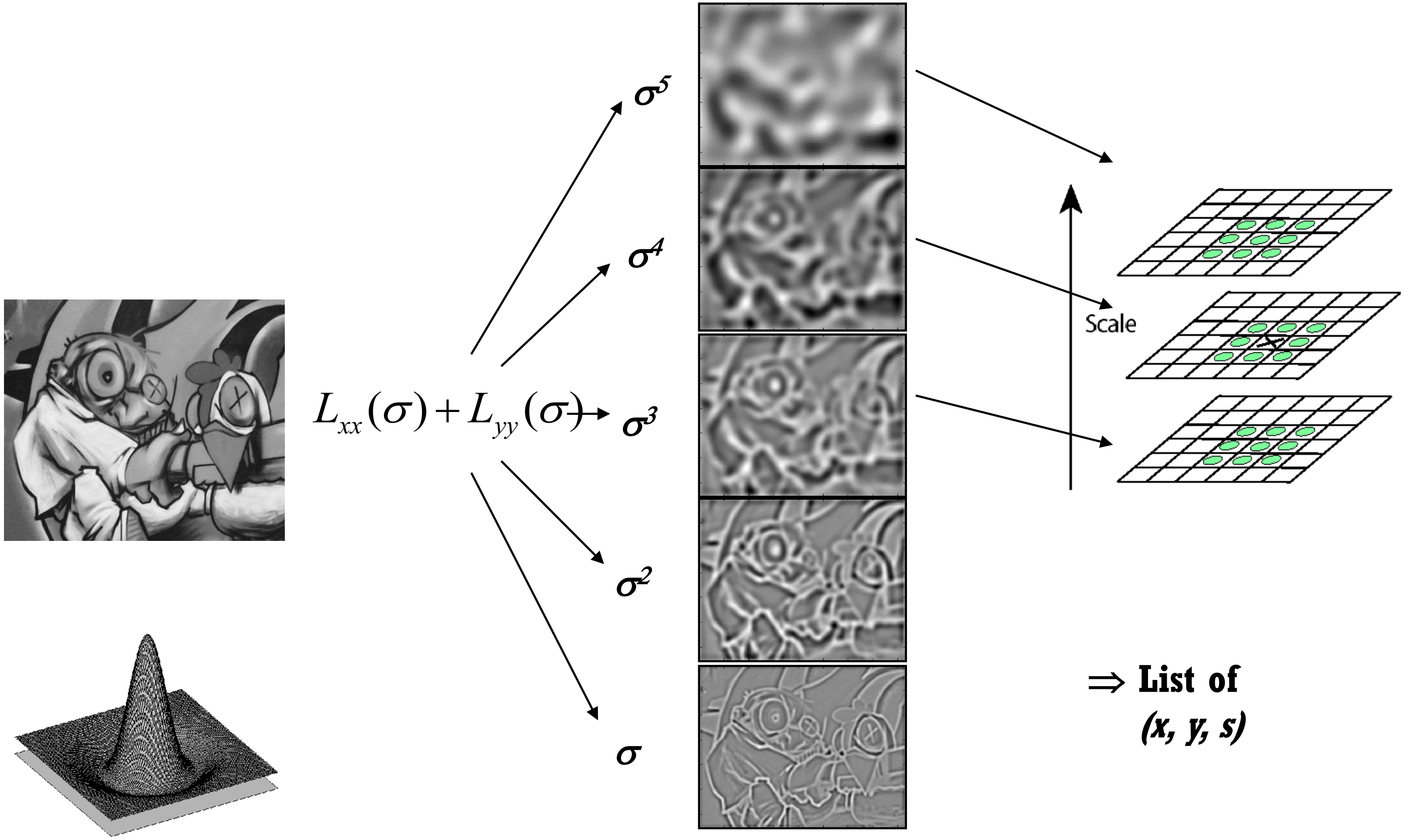
An octave corresponds to doubling the value of σ

Sampling with step σ^4 =2

Original image



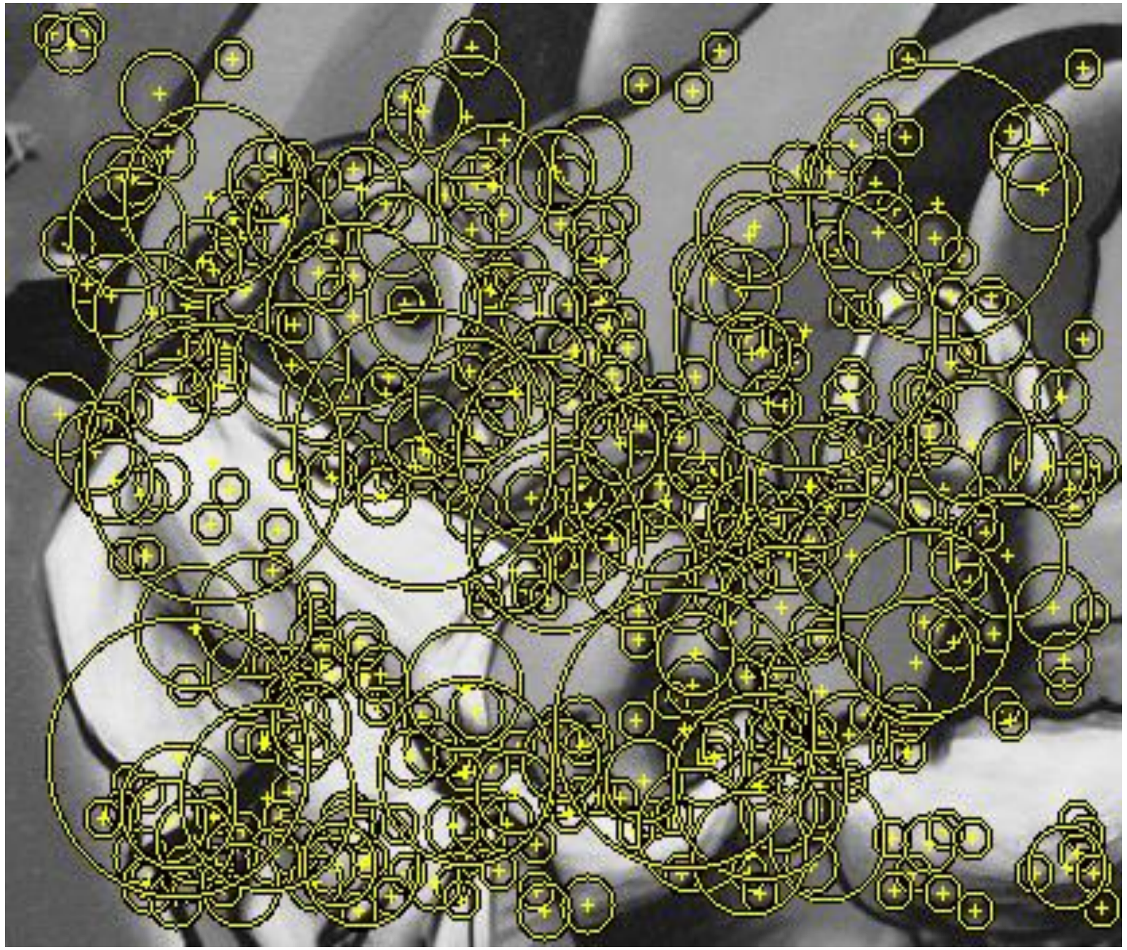
FIND LOCAL MAXIMA IN POSITION-SCALE SPACE OF DIFFERENCE-OF-GAUSSIAN



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

K. Grauman, B. Leibe

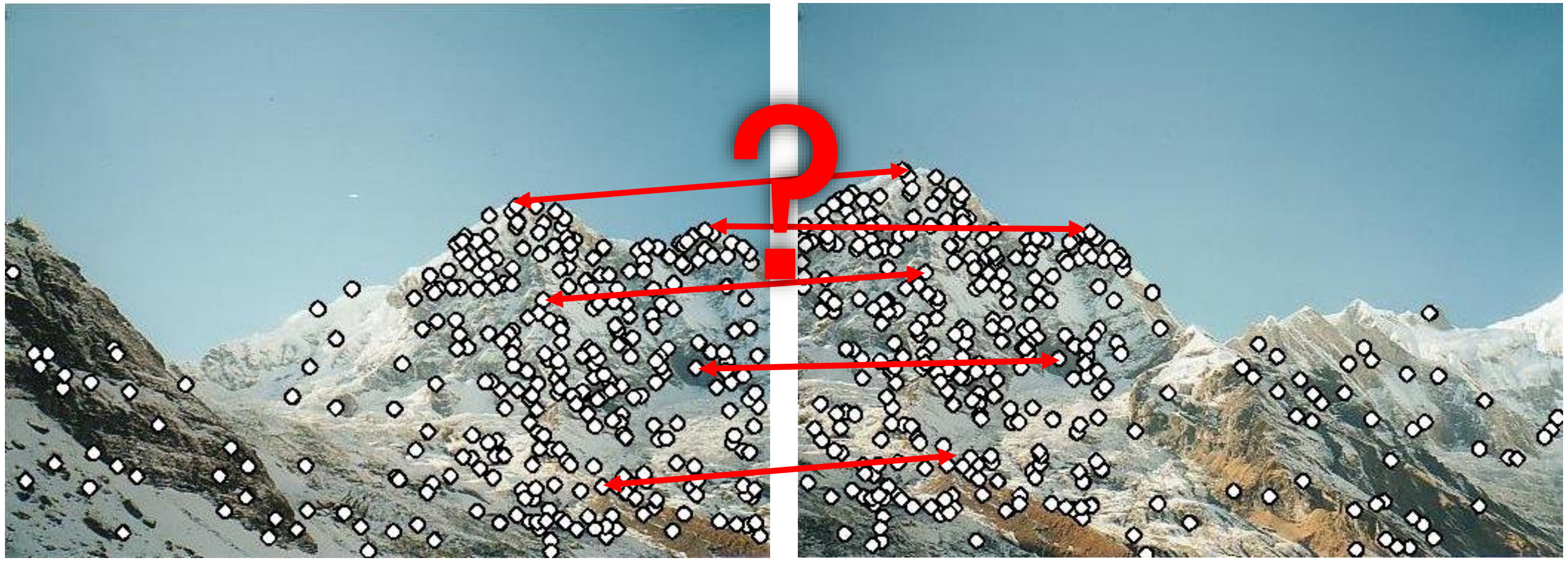
RESULTS: DIFFERENCE-OF-GAUSSIAN



K. Grauman, B. Leibe

FEATURE DESCRIPTORS

We know how to detect good points
Next question: **How to match them?**



Answer: Come up with a *descriptor* for each point, find similar descriptors between the two images

CHALLENGES

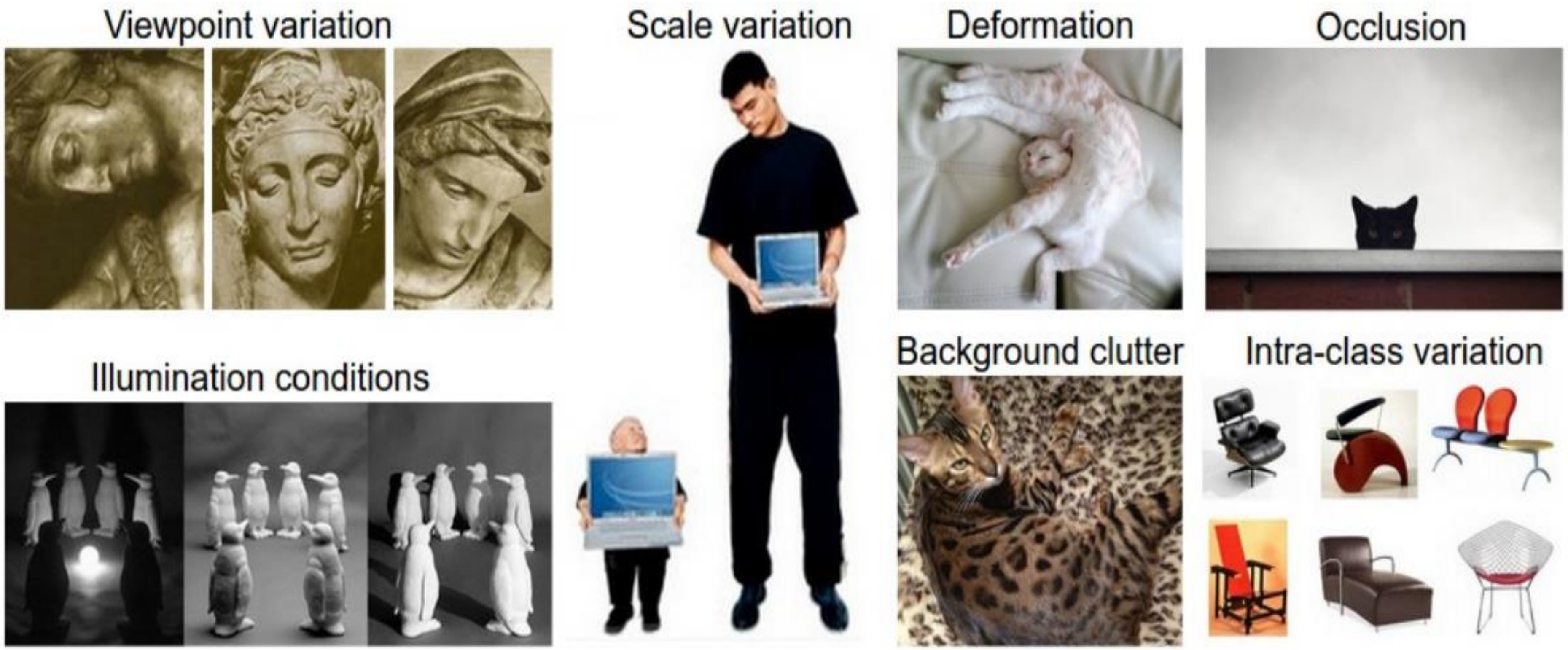


Image Source: cs231n, Oxford University

Image descriptor

- Descriptions of the visual features
- Described by appearance based characteristics such as color, shape, etc.

Image descriptor

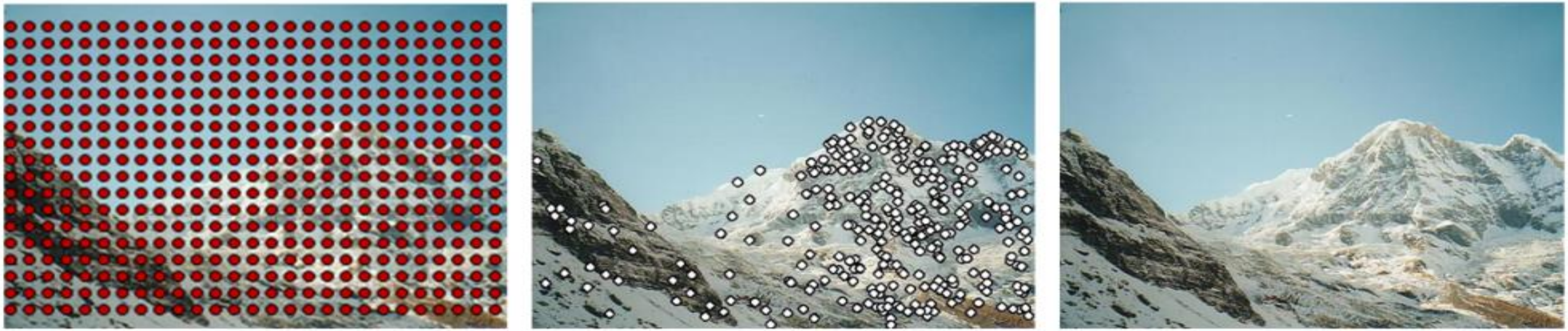
- Descriptions of the visual features
- Described by appearance based characteristics such as color, shape, etc.

A descriptor must be

- Distinctive
- Robust
- Compact
- Low Dimensional

Where to compute the descriptors?

- Over interest regions.
- Interest region may be
  - Grid based, Key-Points or Global based.



LOCAL DESCRIPTORS

- Most available descriptors focus on -
  - Edge/gradient information
  - Capture texture information
  - Exploit local relationship
  - Color also play a vital role
  - Shape features
  - Feature fusion

WIDELY USED LOCAL DESCRIPTORS

- SIFT – Scale Invariant Feature Transform

Distinctive image **features** from **scale-invariant** keypoints
DG Lowe - International journal of computer vision, 2004 - Springer
… the assigned orientation, **scale**, and loca- tion for each **feature**, thereby providing **invariance** to these … **Invariant** Fea- ture **Transform** (SIFT), as it **transforms** image data into **scale-invariant** coordinates relative … that densely cover the image over the full range of **scales** and locations …
Cited by 50252 Related articles All 179 versions

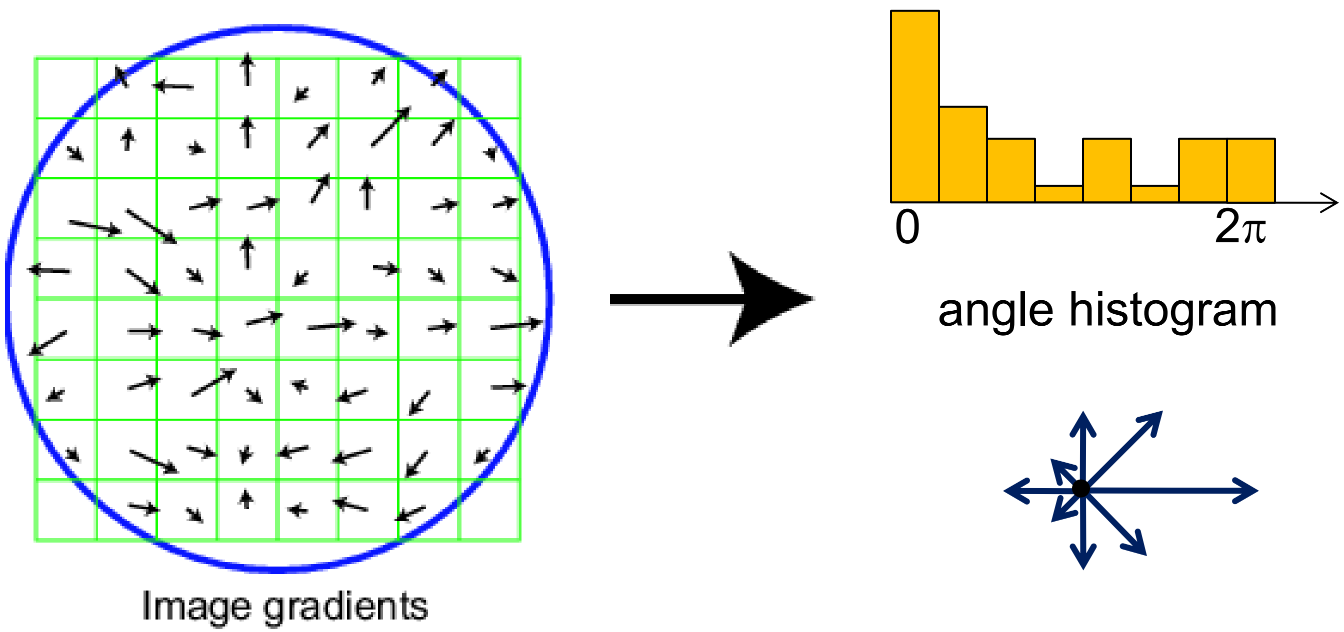
- LBP – Local Binary Pattern

Multiresolution gray-scale and rotation invariant texture classification with **local binary patterns**
T Ojala, M Pietikainen… - … Transactions on **pattern** …, 2002 - ieeexplore.ieee.org
Presents a theoretically very simple, yet efficient, multiresolution approach to gray-scale and rotation invariant texture classification based on **local binary patterns** and nonparametric discrimination of sample and prototype distributions. The method is based on recognizing …
Cited by 12251 Related articles All 16 versions

Scale Invariant Feature Transform (SIFT)

Basic idea:

- Take 16x16 square window around detected feature
- Compute edge orientation for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations

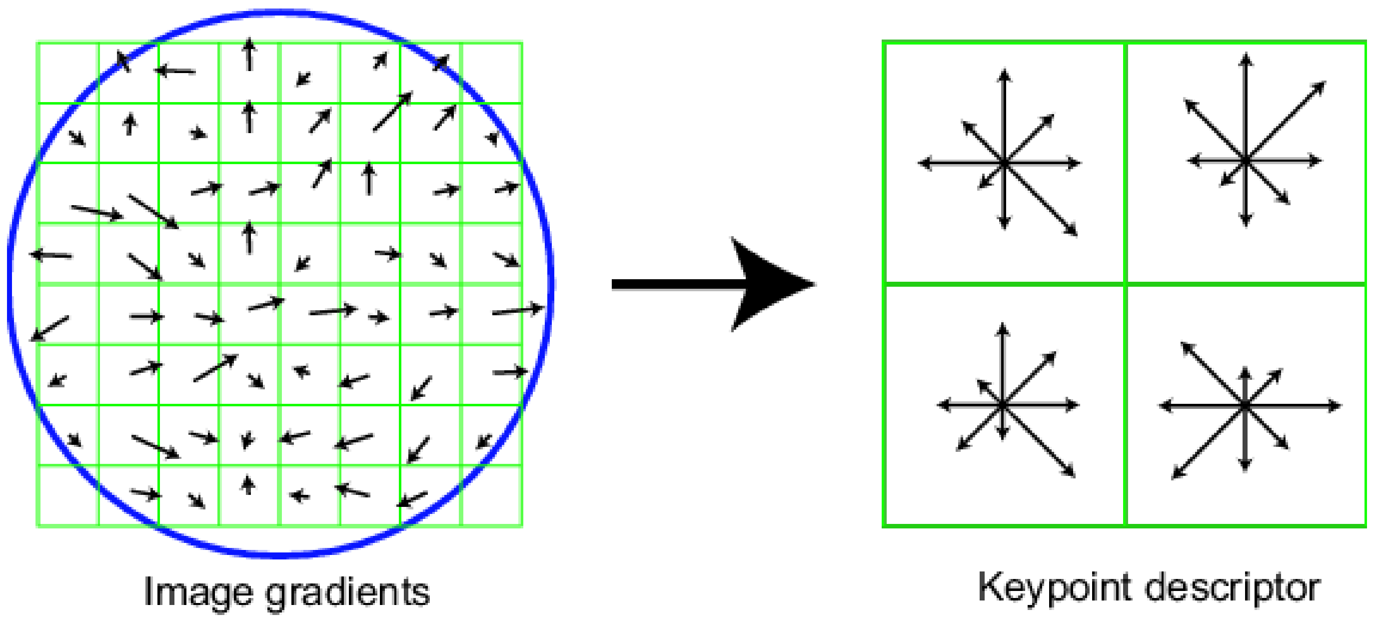


Adapted from slide by David Lowe

SIFT DESCRIPTOR

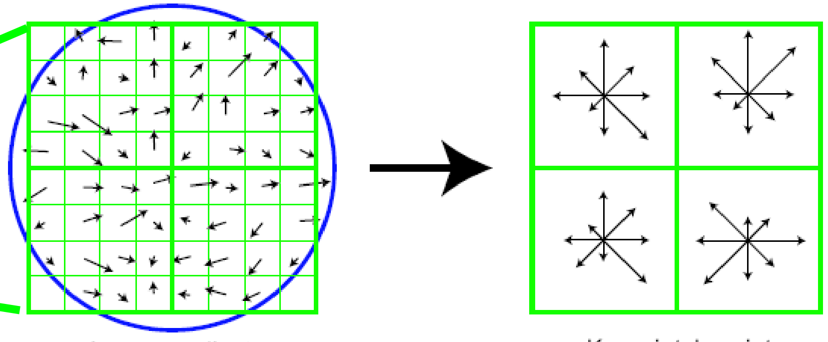
Full version

- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor



Adapted from slide by David Lowe

LOCAL DESCRIPTORS: SIFT DESCRIPTOR



Histogram of oriented gradients

- **Captures important texture information**
- **Robust to small translations / affine deformations**

[Lowe, ICCV 1999]

K. Grauman, B. Leibe

FEATURE MATCHING

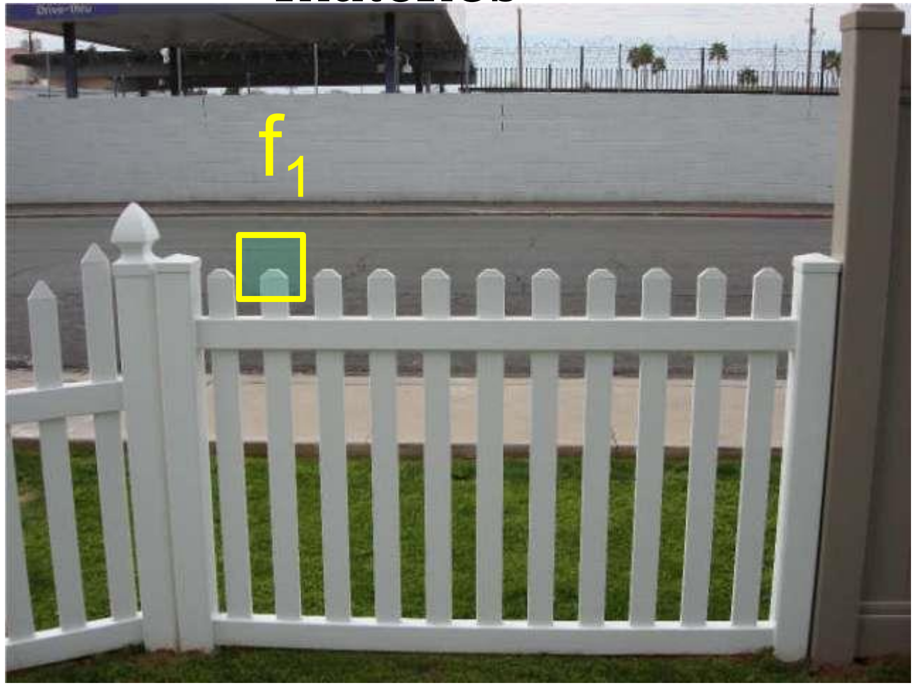
Given a feature in I_1, how to find the best match in I_2?

1. Define distance function that compares two descriptors
2. Test all the features in I_2, find the one with min distance

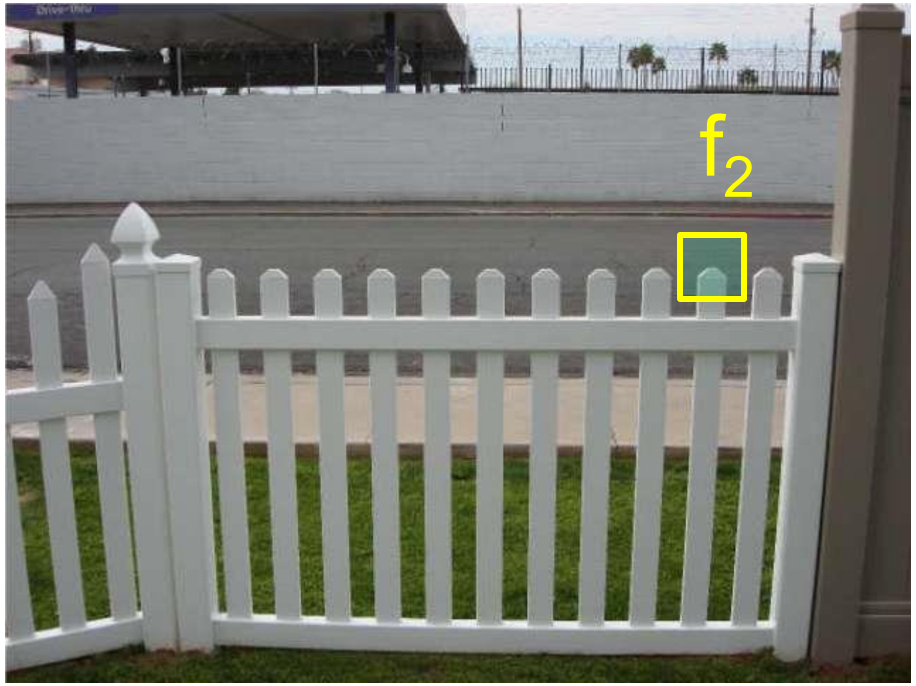
FEATURE DISTANCE

How to define the difference between two features f_1, f_2?

- Simple approach: L_2 distance, $||f_1 - f_2||$
- can give good scores to ambiguous (incorrect) matches



I_1

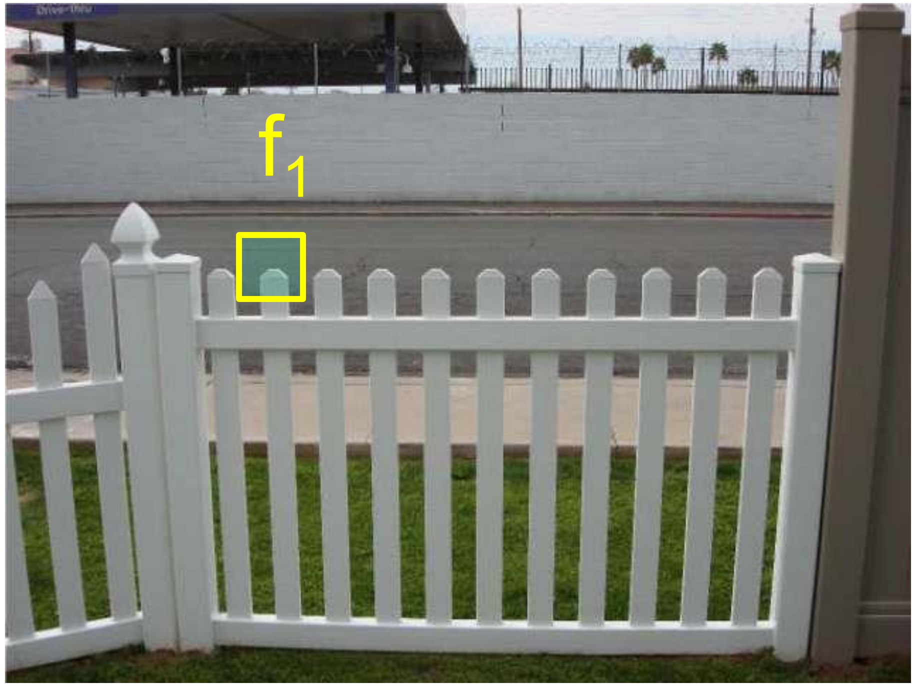


I_2

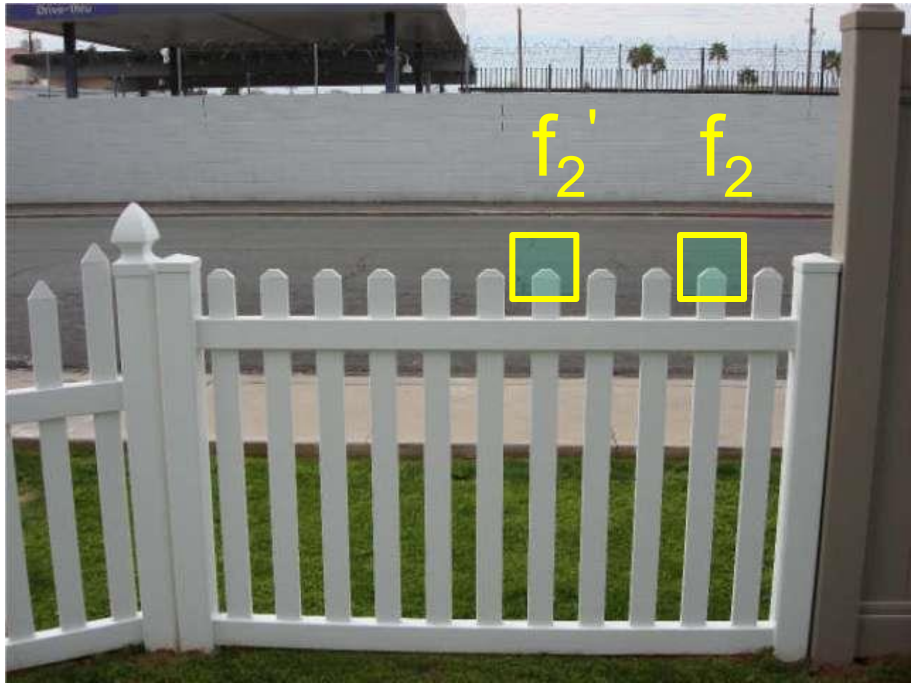
FEATURE DISTANCE

How to define the difference between two features f_1, f_2?

- Better approach: ratio distance = $||f_1 - f_2|| / || f_1 - f_2' ||$
  - f_2 is best SSD match to f_1 in I_2
  - f_2' is 2^{nd} best SSD match to f_1 in I_2
  - gives large values for ambiguous matches

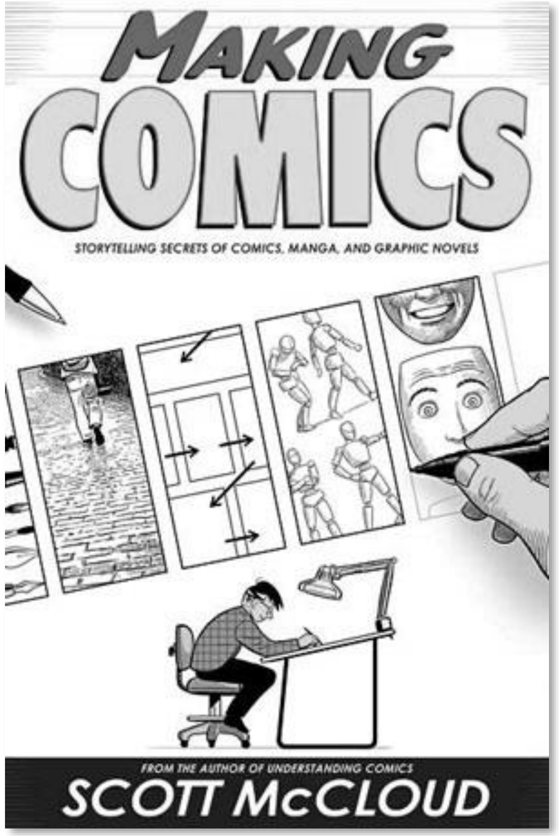


I_1

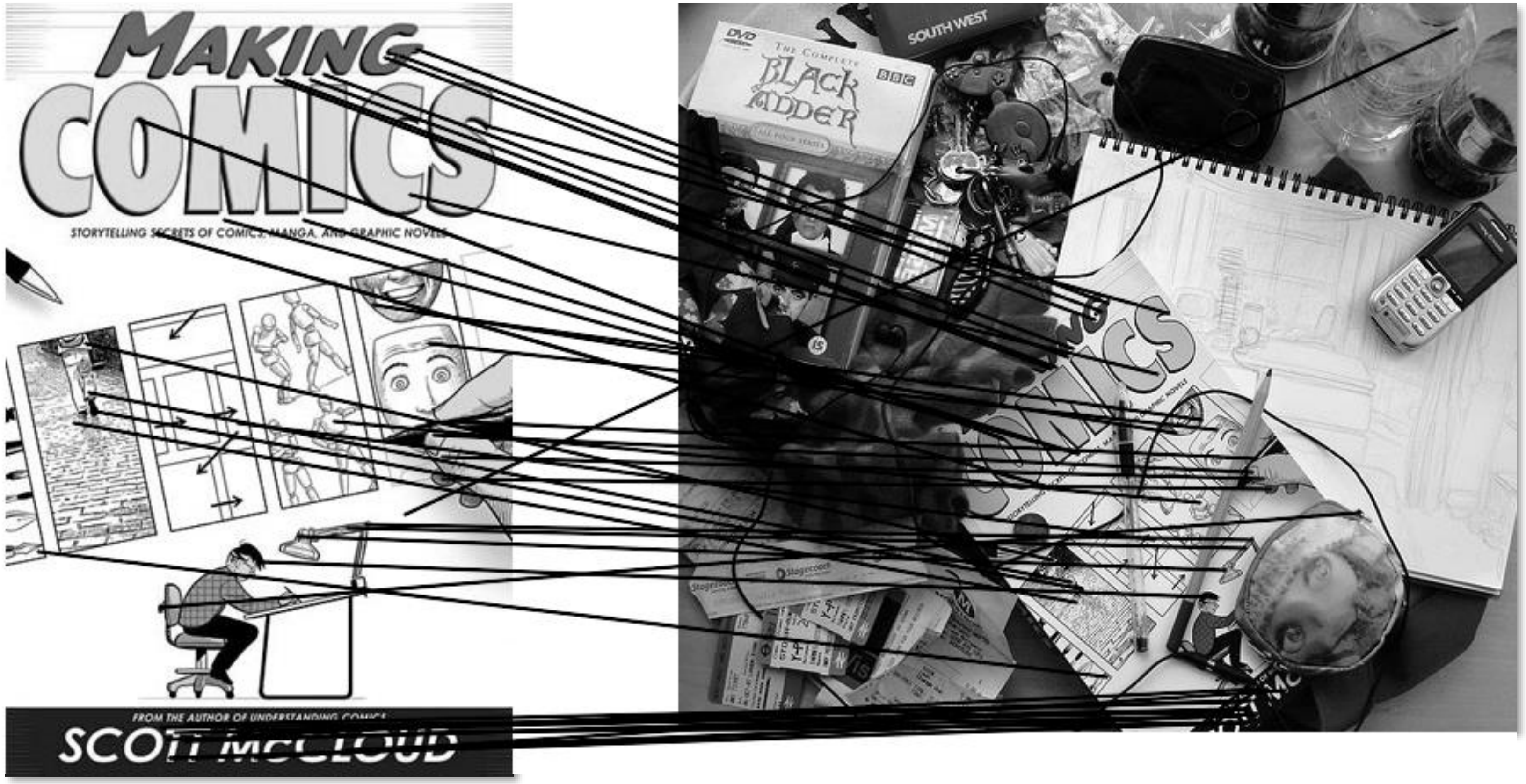


I_2

FEATURE MATCHING EXAMPLE

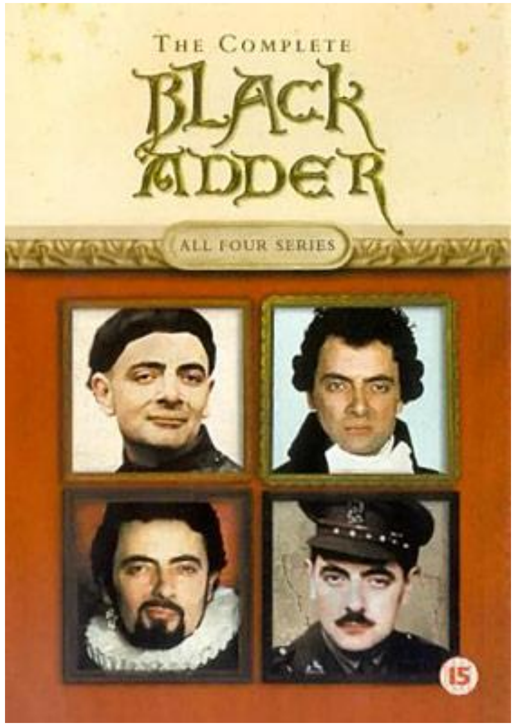


FEATURE MATCHING EXAMPLE

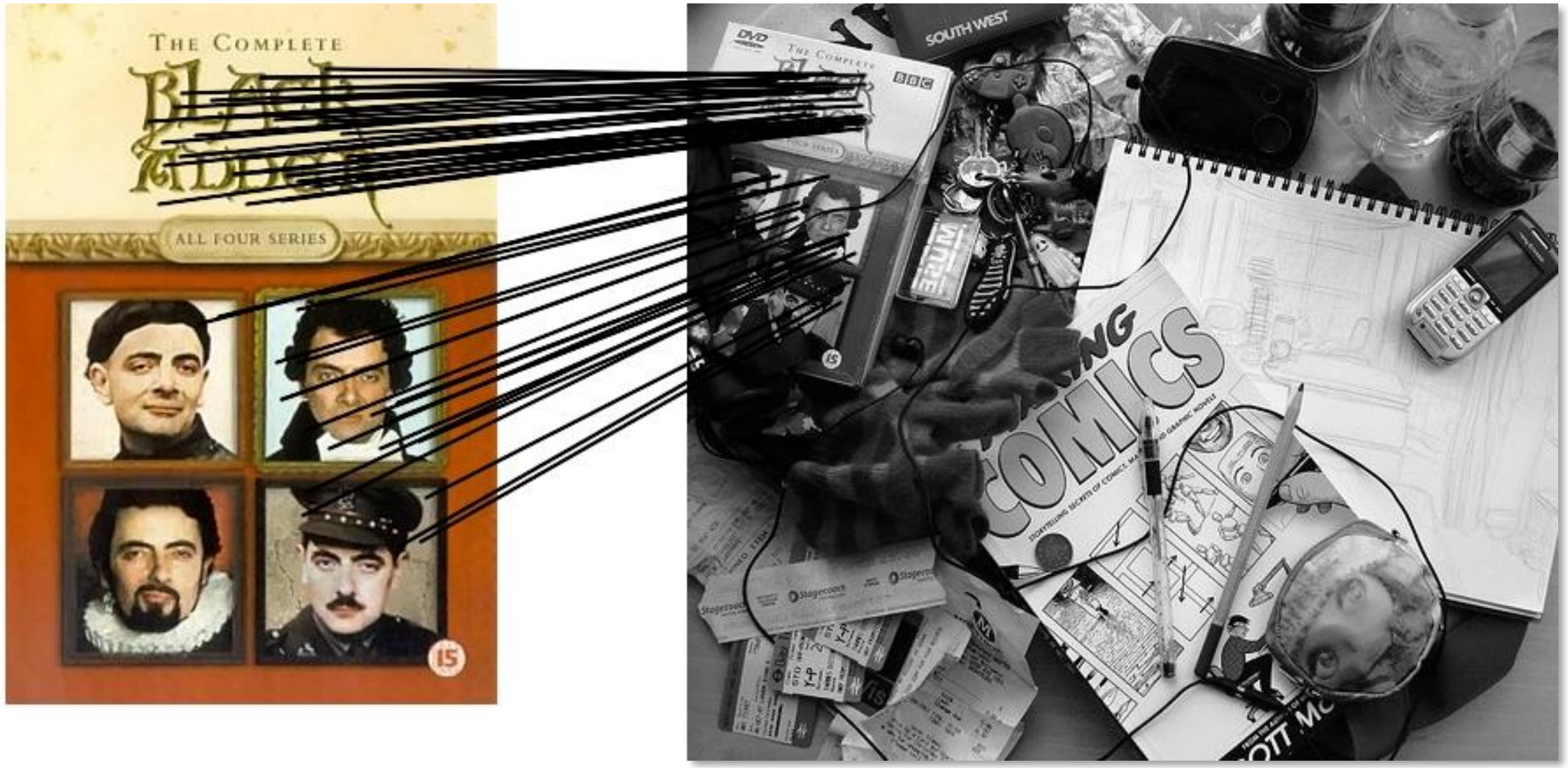


51 matches

FEATURE MATCHING EXAMPLE



FEATURE MATCHING EXAMPLE



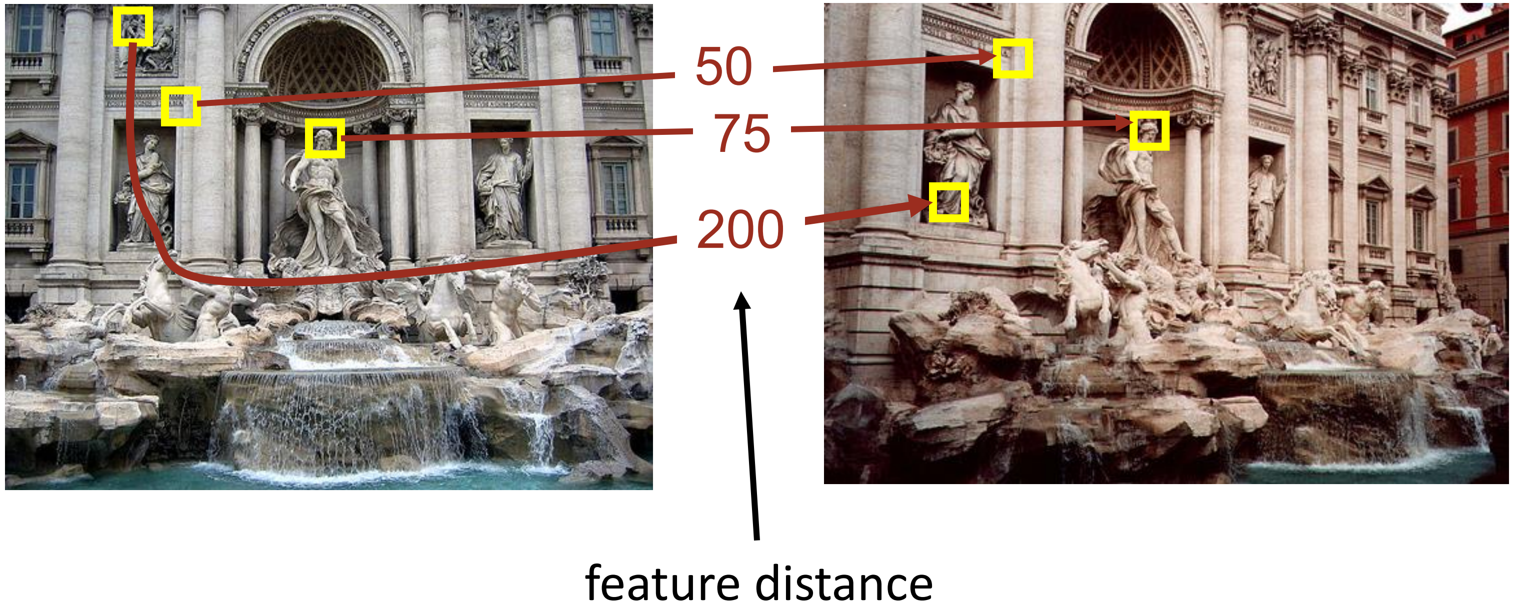
58 matches

EVALUATING THE RESULTS

How can we measure the performance of a feature matcher?

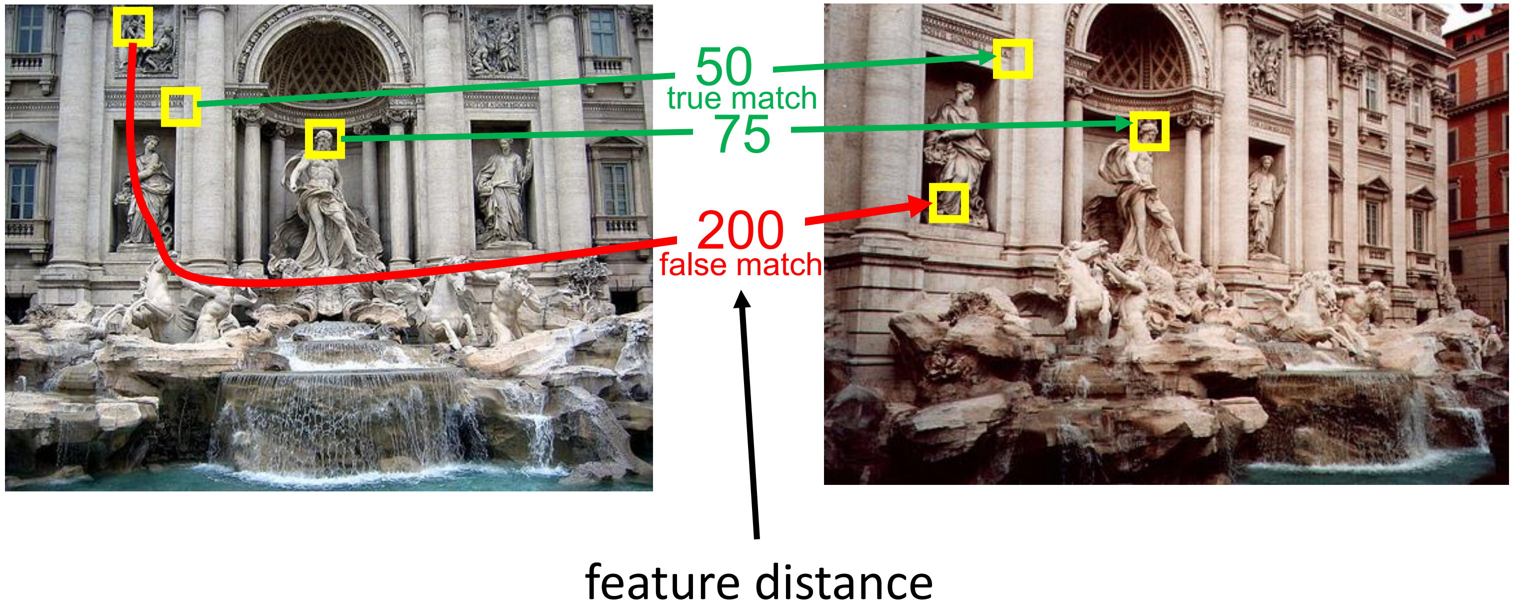
EVALUATING THE RESULTS

How can we measure the performance of a feature matcher?



TRUE/FALSE POSITIVES

How can we measure the performance of a feature matcher?



The distance threshold affects performance

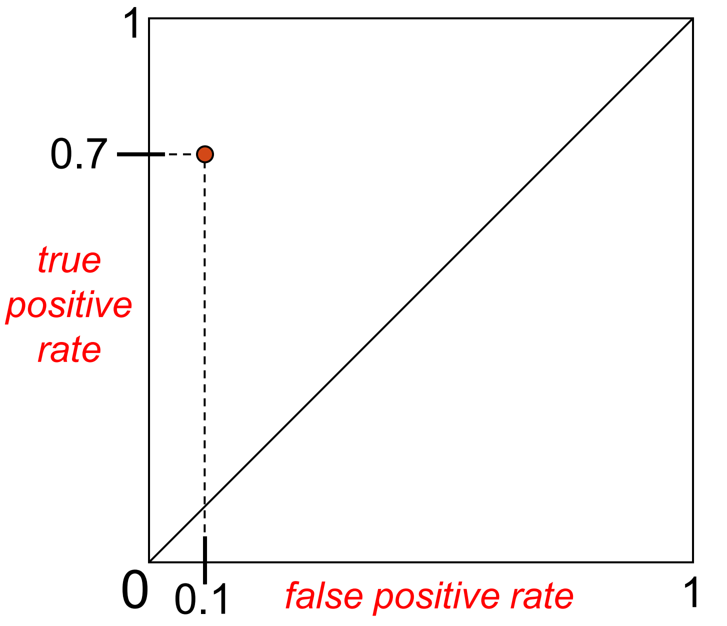
- True positives = # of detected matches that are correct
  - Suppose we want to maximize these—how to choose threshold?
- False positives = # of detected matches that are incorrect
  - Suppose we want to minimize these—how to choose threshold?

EVALUATING THE RESULTS

How can we measure the performance of a feature matcher?

$$\frac{\text{\# true positives}}{\text{\# correctly matched features (positives)}}$$

"recall"



$$\frac{\text{\# false positives}}{\text{\# incorrectly matched features (negatives)}}$$

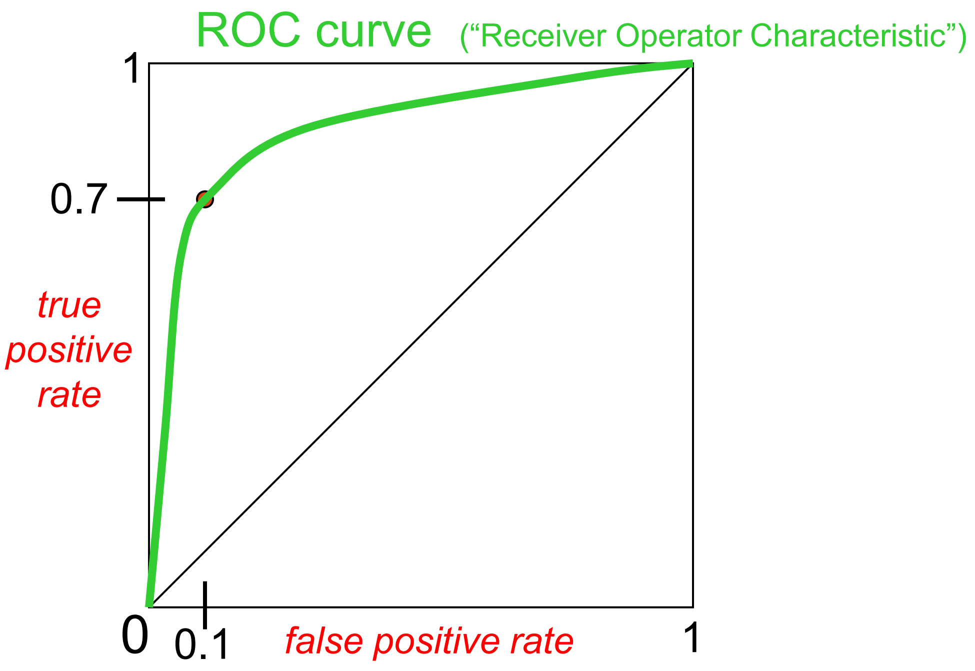
1 - "precision"

EVALUATING THE RESULTS

How can we measure the performance of a feature matcher?

$$\frac{\text{\# true positives}}{\text{\# correctly matched features (positives)}}$$

"recall"



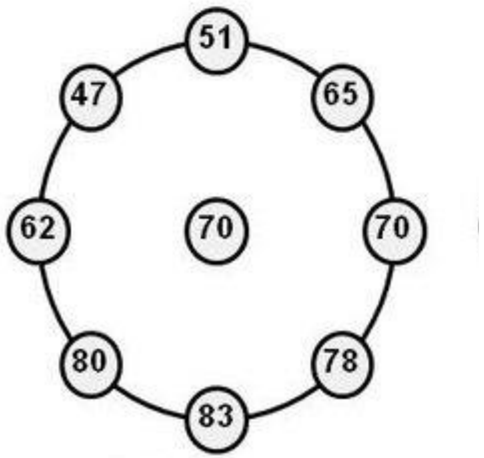
$$\frac{\text{\# false positives}}{\text{\# incorrectly matched features (negatives)}}$$

1 - "precision"

Variations of SIFT features

- PCA-SIFT
- SURF
- GLOH
- Spin Image
- And Many More

Local Binary Pattern (LBP)



1. Sample

Image Source: scholarpedia

[IEEE TPAMI 2002]

Local Binary Pattern (LBP)

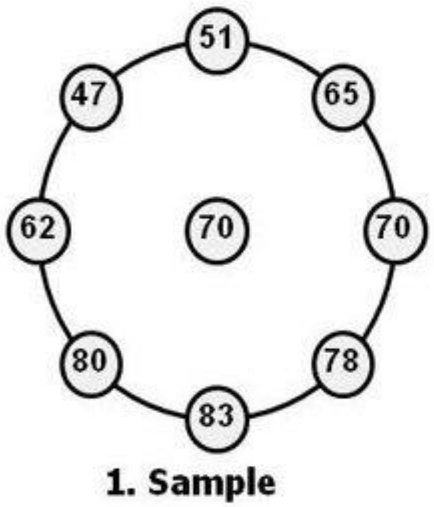


Image Source: scholarpedia

[IEEE TPAMI 2002]

Local Binary Pattern (LBP)

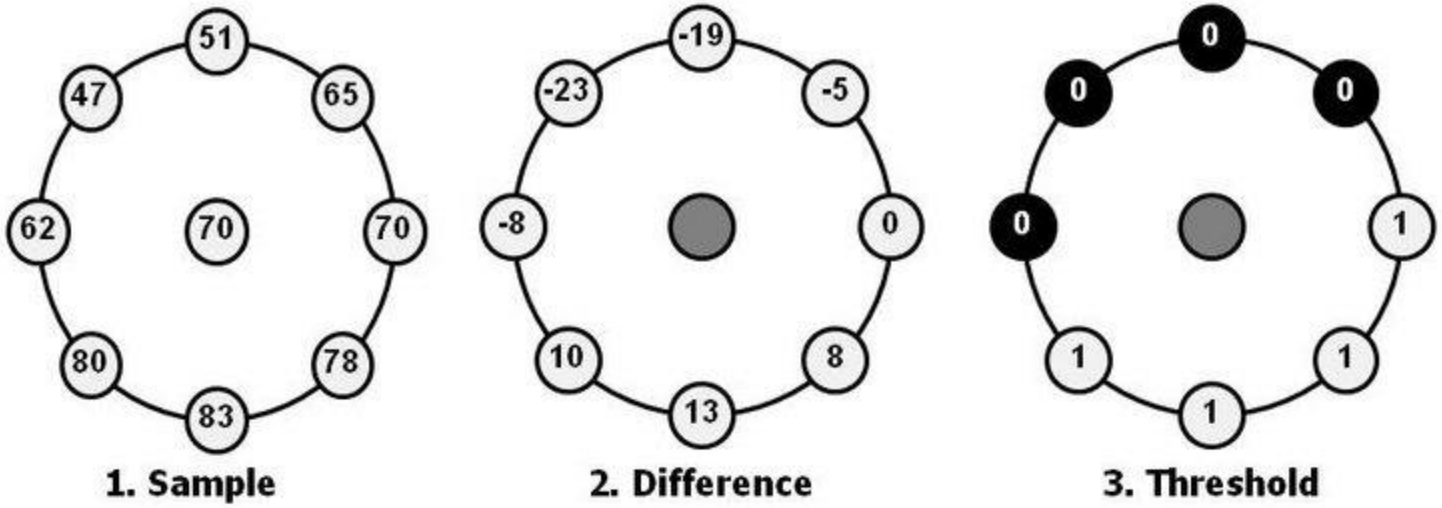


Image Source: scholarpedia

[IEEE TPAMI 2002]

Local Binary Pattern (LBP)

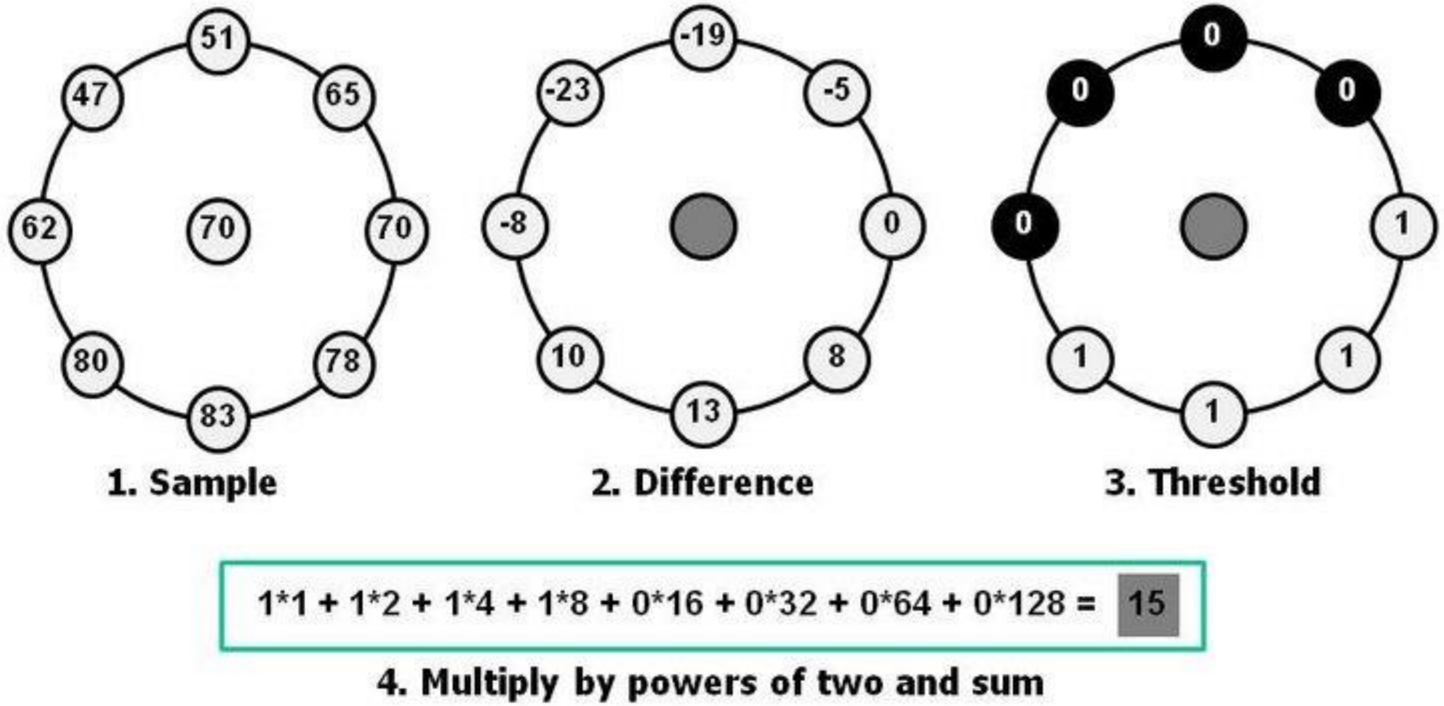


Image Source: scholarpedia

[IEEE TPAMI 2002]

Local Binary Pattern (LBP)

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

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

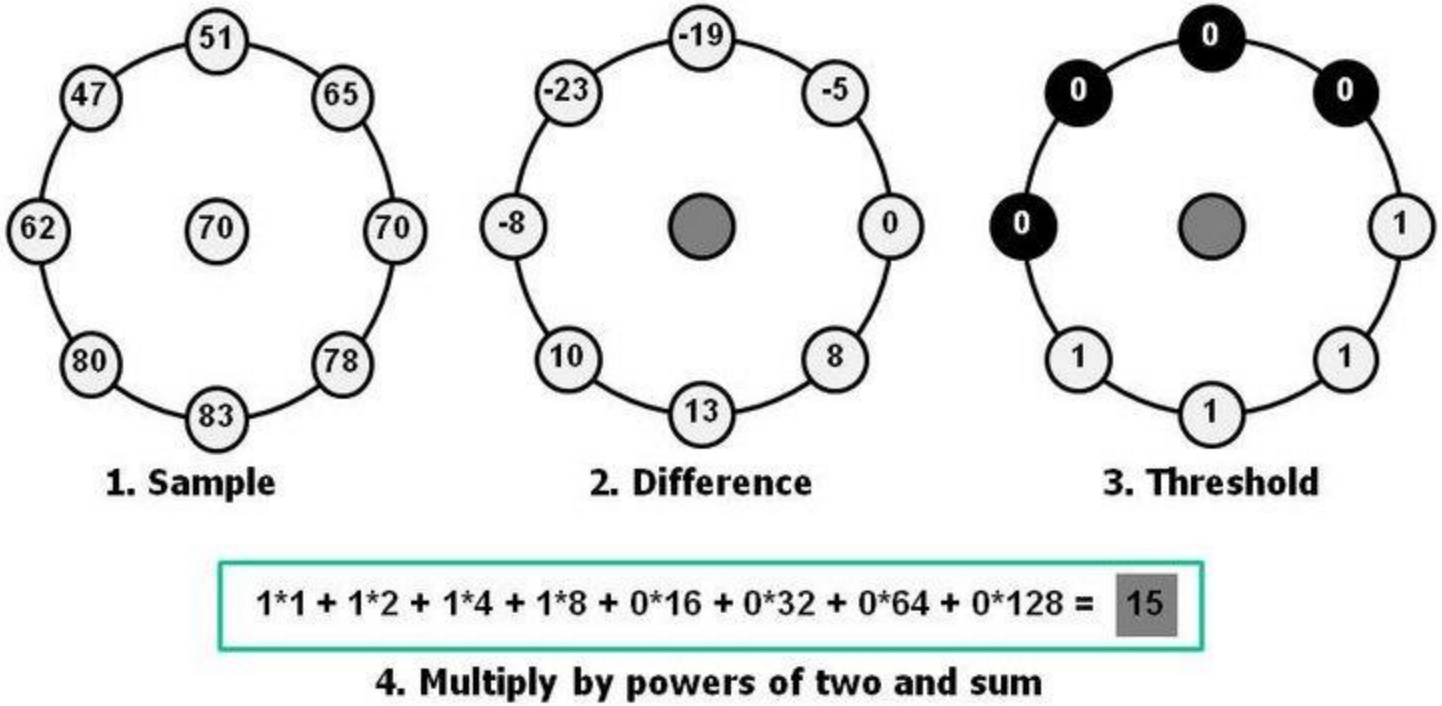


Image Source: scholarpedia

[IEEE TPAMI 2002]

Local Binary Pattern (LBP)

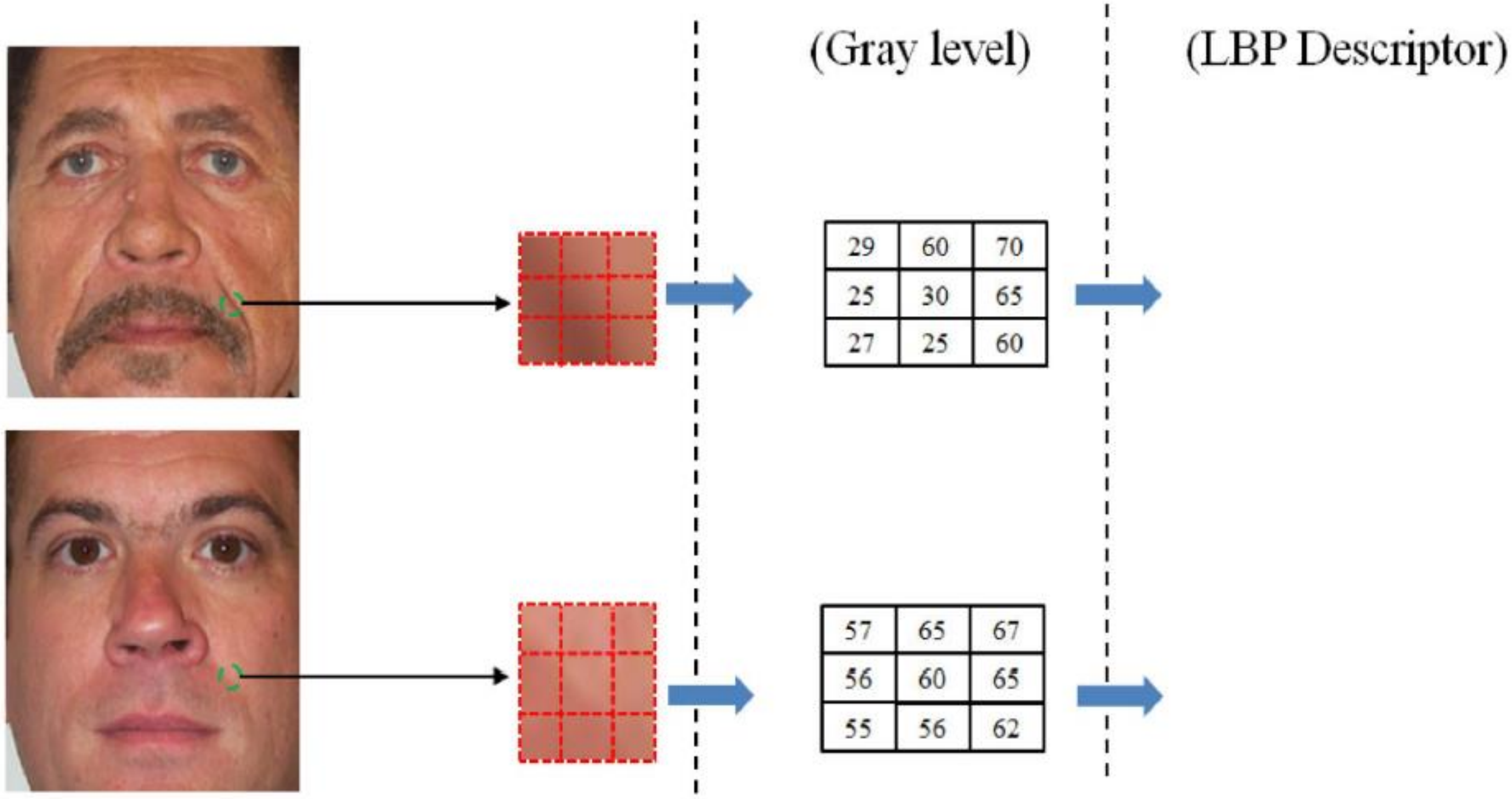


Image Source: Nguyen, D. T., Cho, S. R., & Park, K. R. (2015). Age Estimation-Based Soft Biometrics Considering Optical Blurring Based on Symmetrical Sub-Blocks for MLBP. *Symmetry*, *7*(4), 1882-1913.

Local Binary Pattern (LBP)

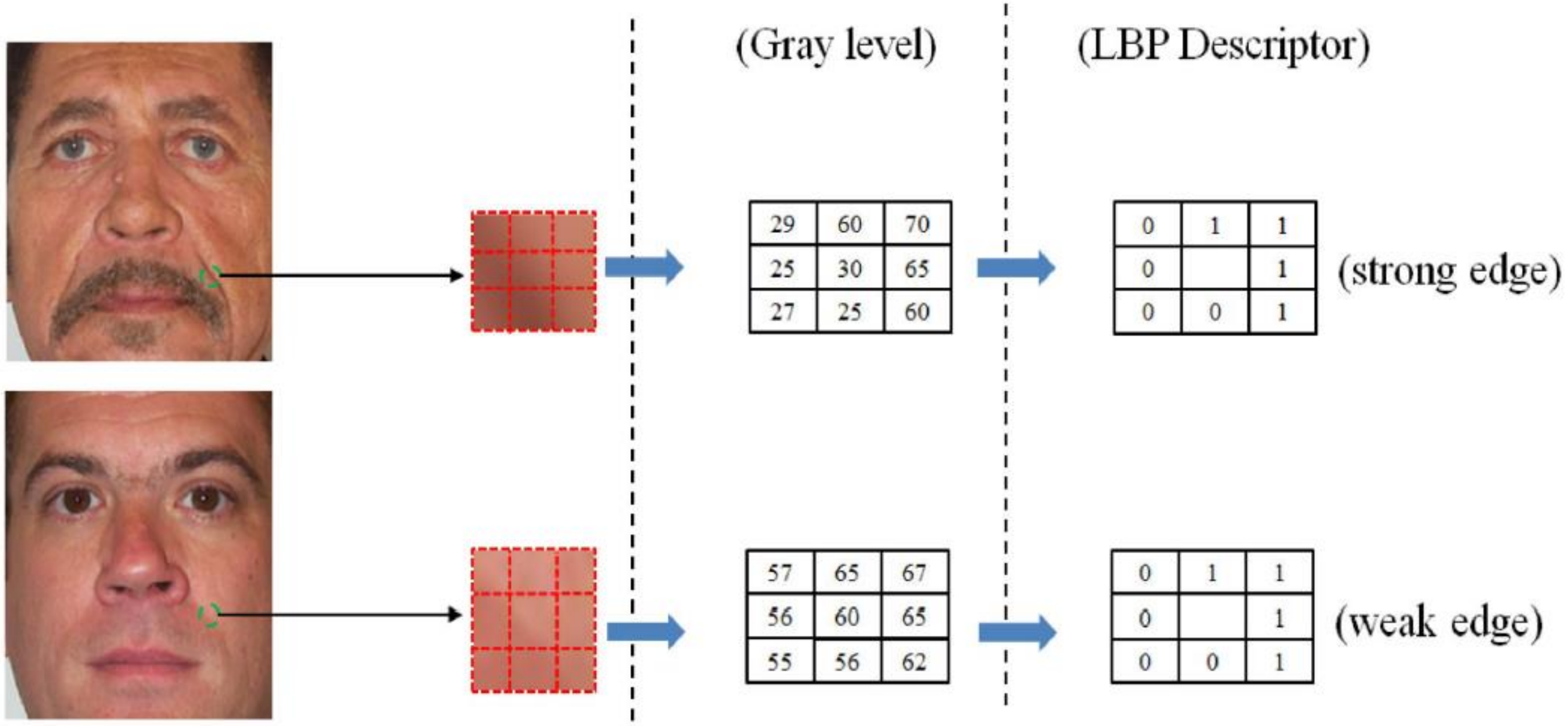


Image Source: Nguyen, D. T., Cho, S. R., & Park, K. R. (2015). Age Estimation-Based Soft Biometrics Considering Optical Blurring Based on Symmetrical Sub-Blocks for MLBP. *Symmetry*, *7*(4), 1882-1913.

Local Binary Pattern (LBP)

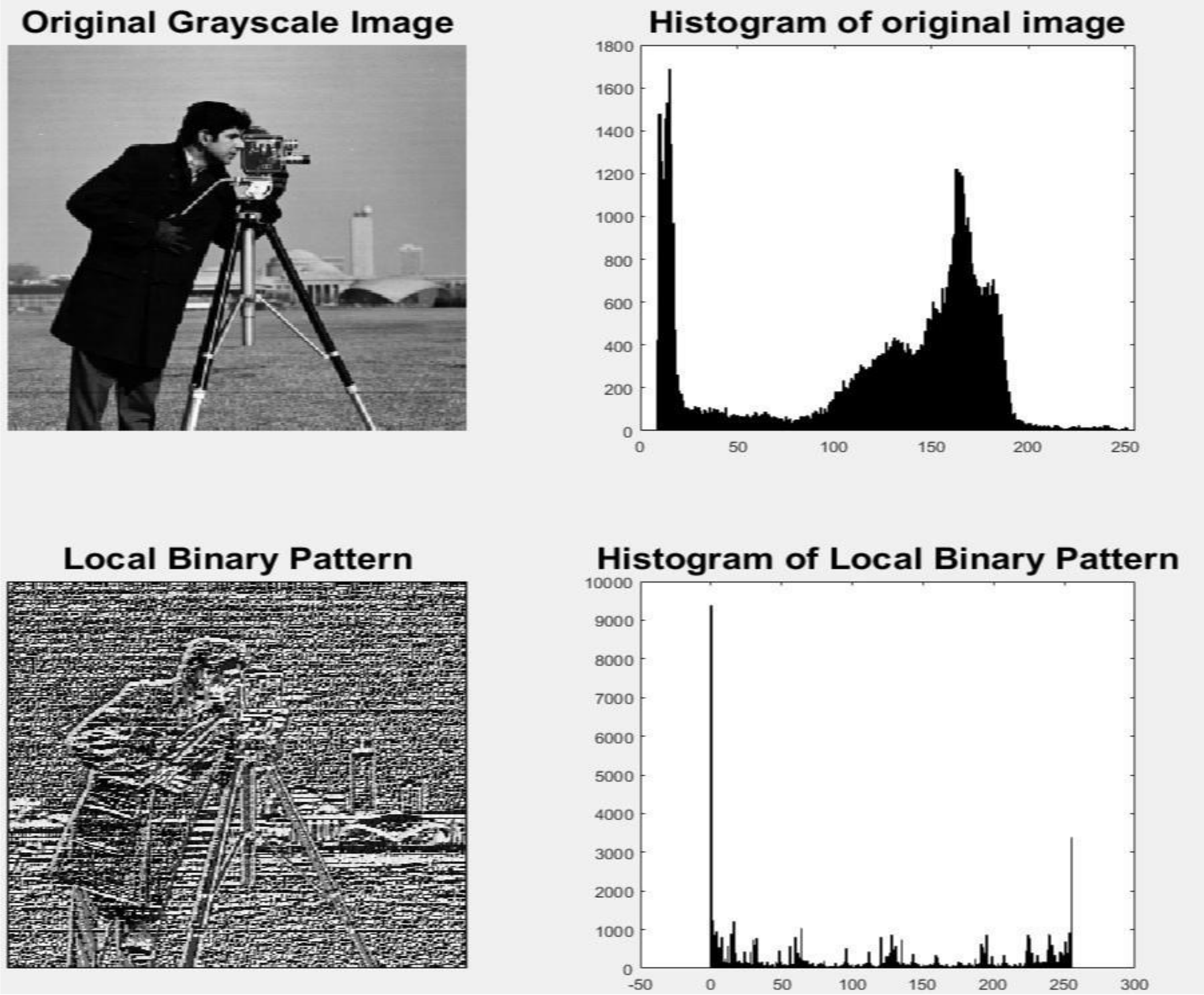


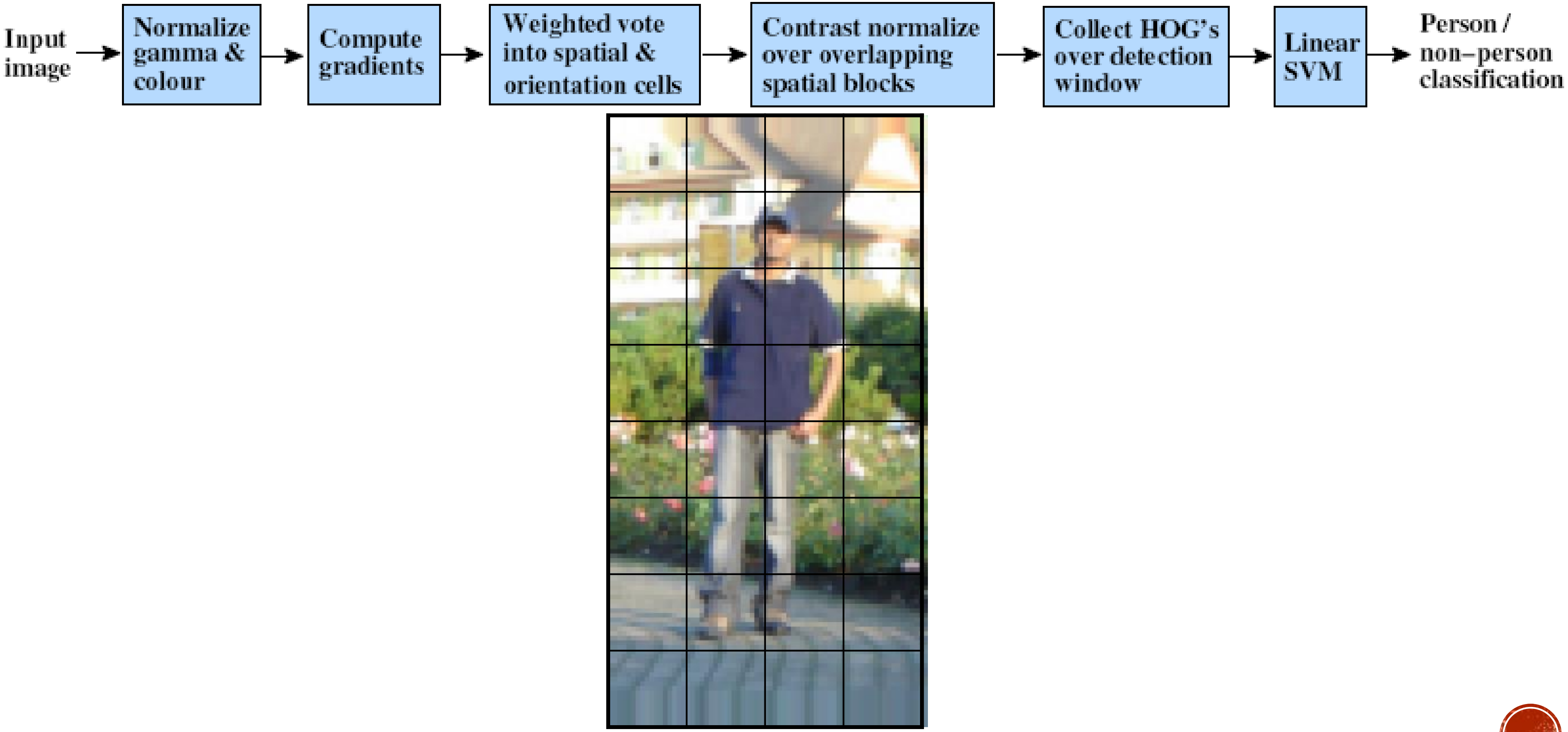
Image Source: Mathworks

DALAL-TRIGGS DETECTOR: HOG

1. Extract fixed-sized (64x128 pixel) window at each position and scale
2. Compute HOG (histogram of gradient) features within each window
3. Score the window with a linear SVM classifier
4. Perform non-maxima suppression to remove overlapping detections with lower scores

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

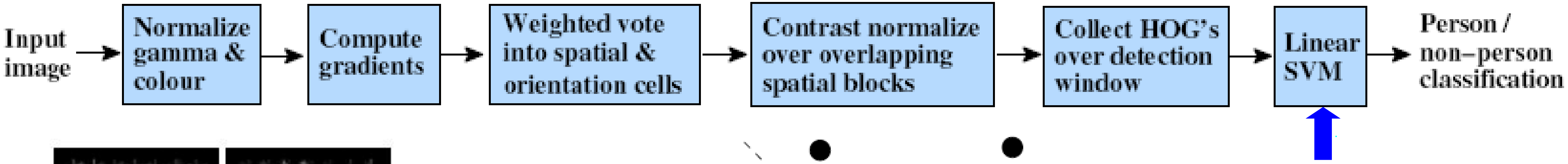
HISTOGRAM OF GRADIENTS (HOG)

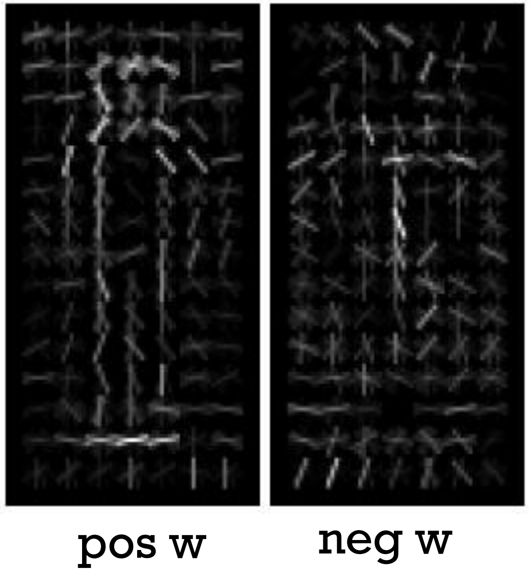


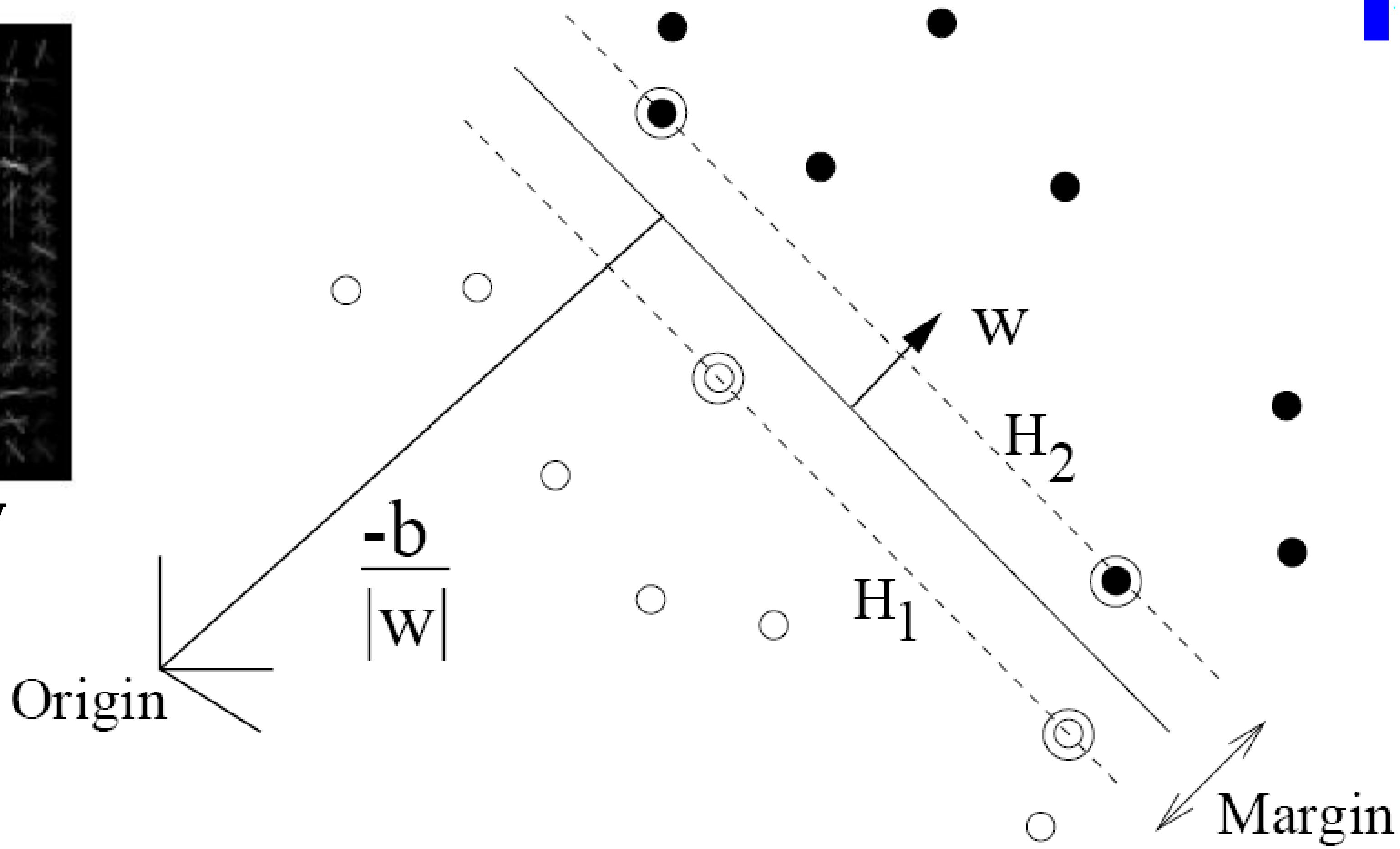
Slides by Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

HISTOGRAM OF GRADIENTS (HOG)



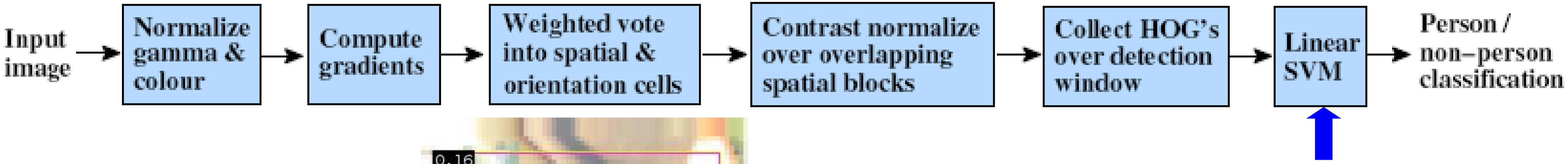




Slides by Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

HISTOGRAM OF GRADIENTS (HOG)



$$0.16 = w^T x - b$$

$$sign(0.16) = 1$$

$=>$ pedestrian

Slides by Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

DETECTION EXAMPLES

ACKNOWLEDGEMENT

Thanks to the following courses and corresponding researchers for making their teaching/research material online

- Convolutional Neural Networks for Visual Recognition, Stanford University
- Deep Learning, Stanford University
- Introduction to Deep Learning, University of Illinois at Urbana-Champaign
- Introduction to Deep Learning, Carnegie Mellon University
- Natural Language Processing with Deep Learning, Stanford University
- And Many More Publicly Available Resources ……

Questions?