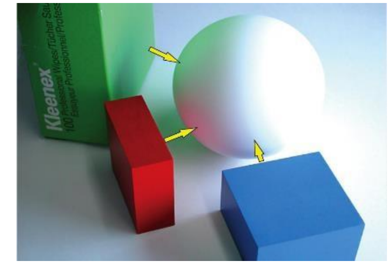
Interest Points

Computer Vision

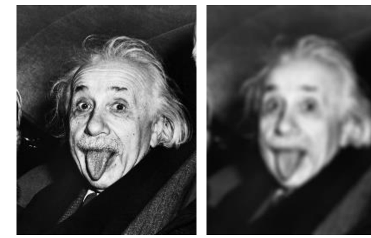
Adduru U G Sankararao, IIIT Sri City

What have we learned so far?

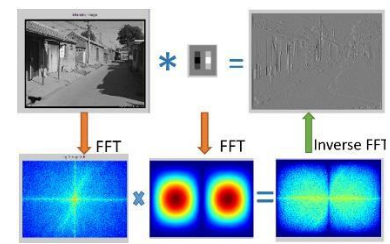
- Light and color
  - What an image records



- Filtering in spatial domain
  - Filtering = weighted sum of neighboring pixels
  - Smoothing, sharpening, measuring texture

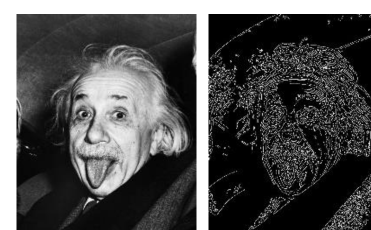


- Filtering in frequency domain
  - Filtering = change frequency of the input image
  - Denoising, sampling, image compression



- Image pyramid (Gaussian and Laplacian)
  - Multi-scale analysis

- Edge detection
  - Canny edge = smooth -> derivative -> thin -> threshold -> link
  - Finding straight lines



Today’s class

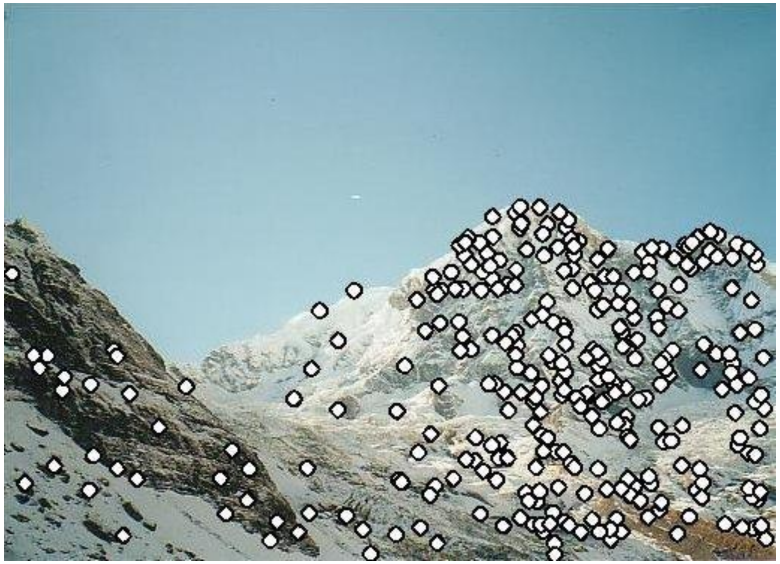
- What is interest point?
- Blob detection
- Corner detection
- Handling scale and orientation

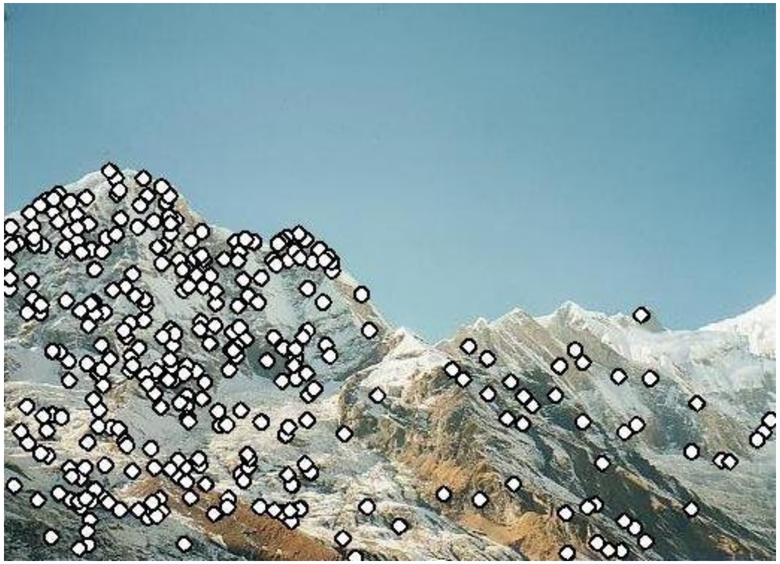
Why extract features?

- 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?

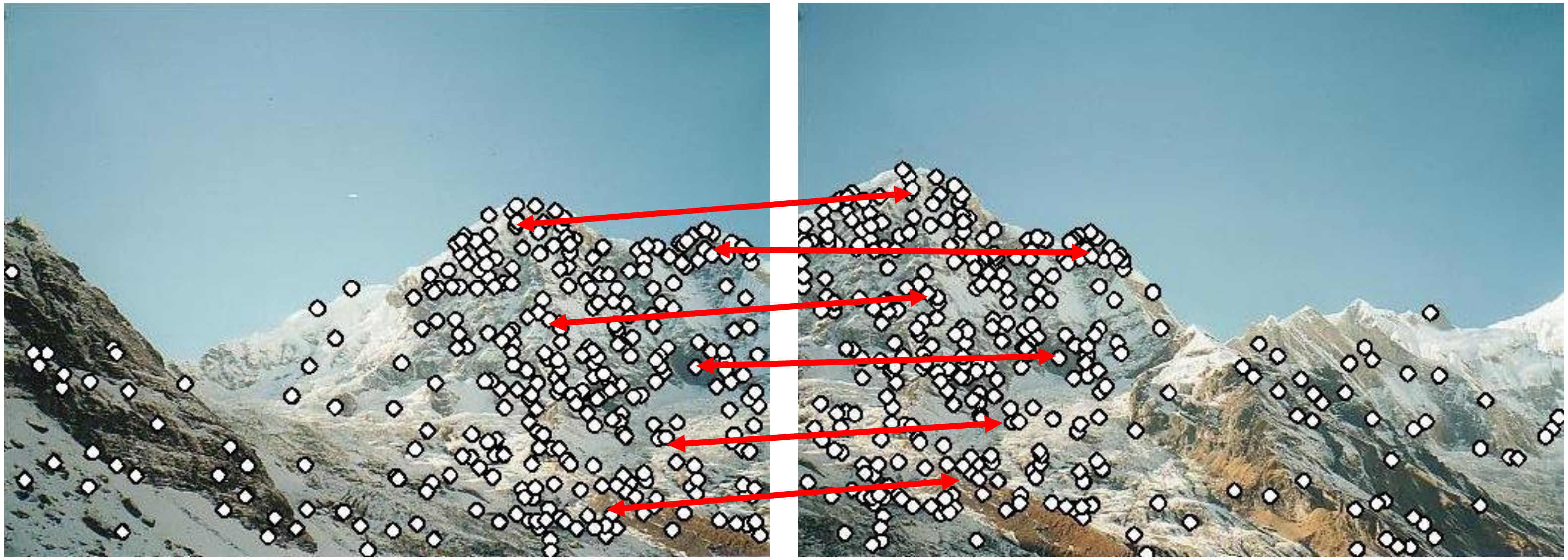




Step 1: extract features

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

Applications

- Key Points/Interest Points are used for:
  - Image alignment
  - 3D reconstruction
  - Motion tracking
  - Object recognition
  - Robot navigation
  - Indexing and database retrieval

Desired Properties of local features

- repeatable: in a transformed image, the same feature is detected at a transformed position

- distinctive: different image features can be discriminated by their local appearance

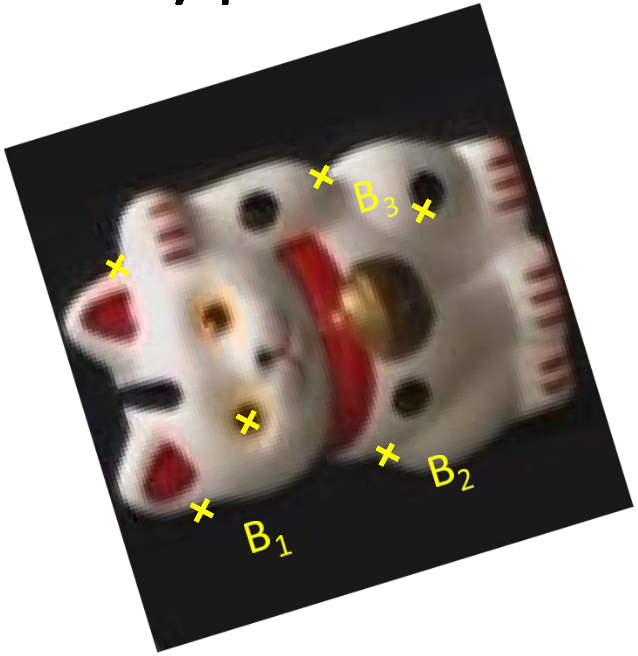
- localized: relatively small regions, robust to occlusion

− elongated: edges, ridges

+ isotropic: blobs, extremal regions

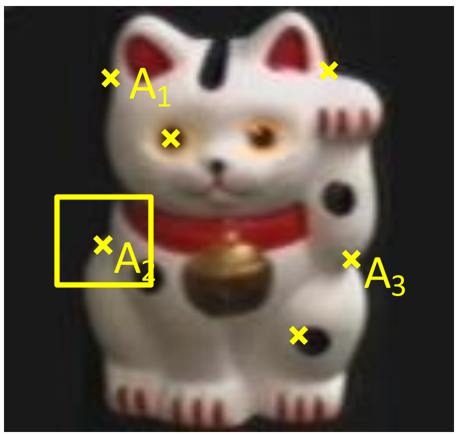
+ points: corners and junctions

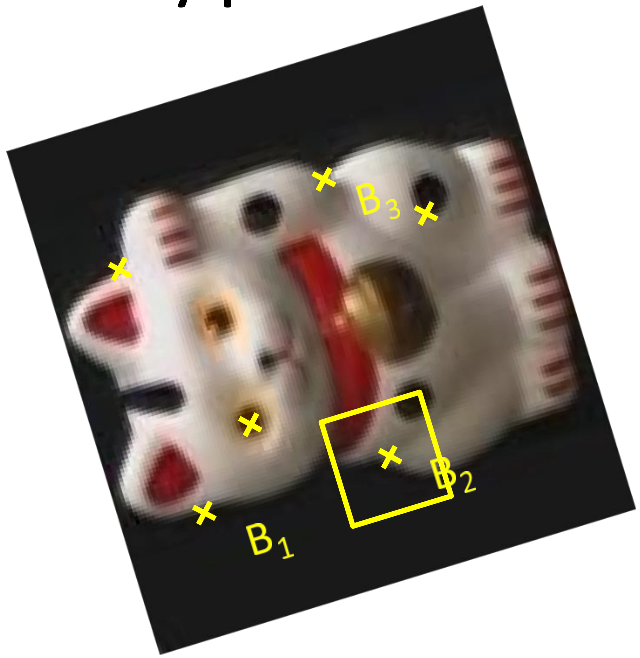
Overview of Keypoint Matching



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

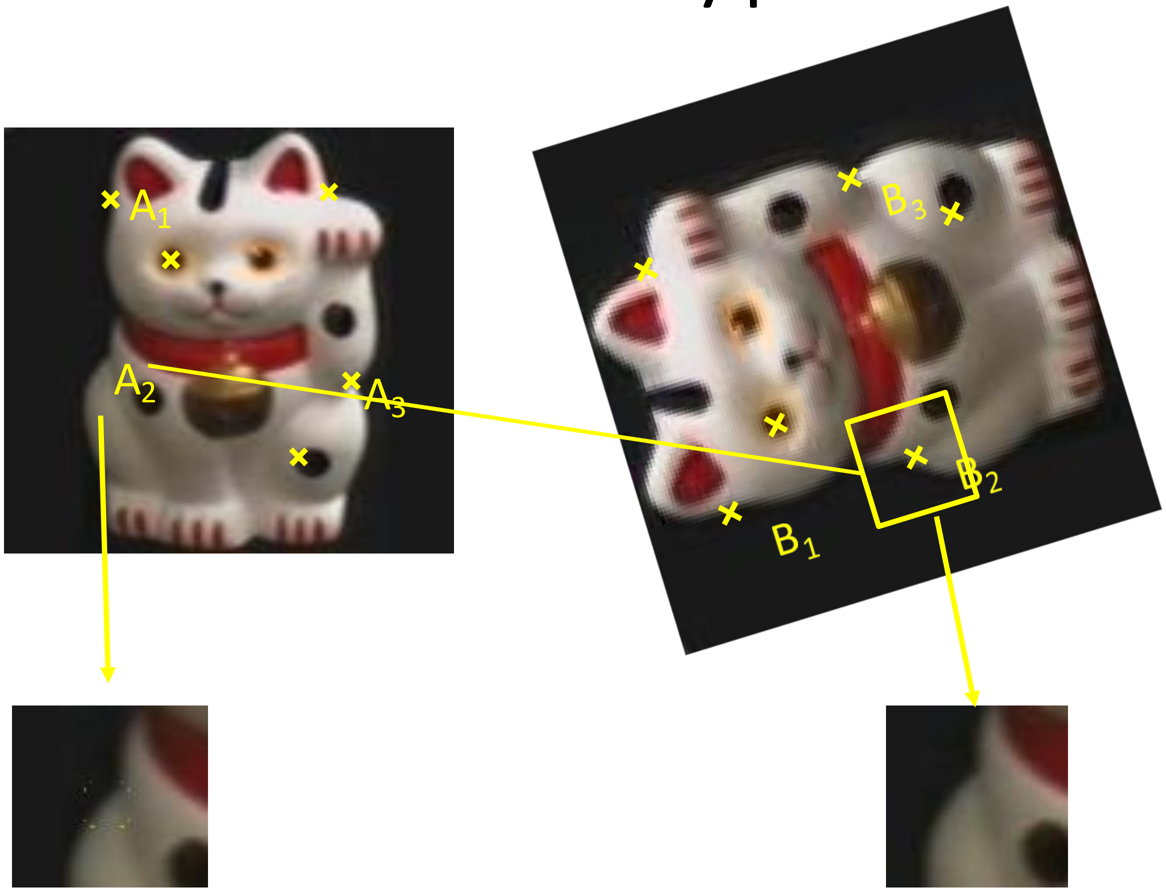
Overview of Keypoint Matching





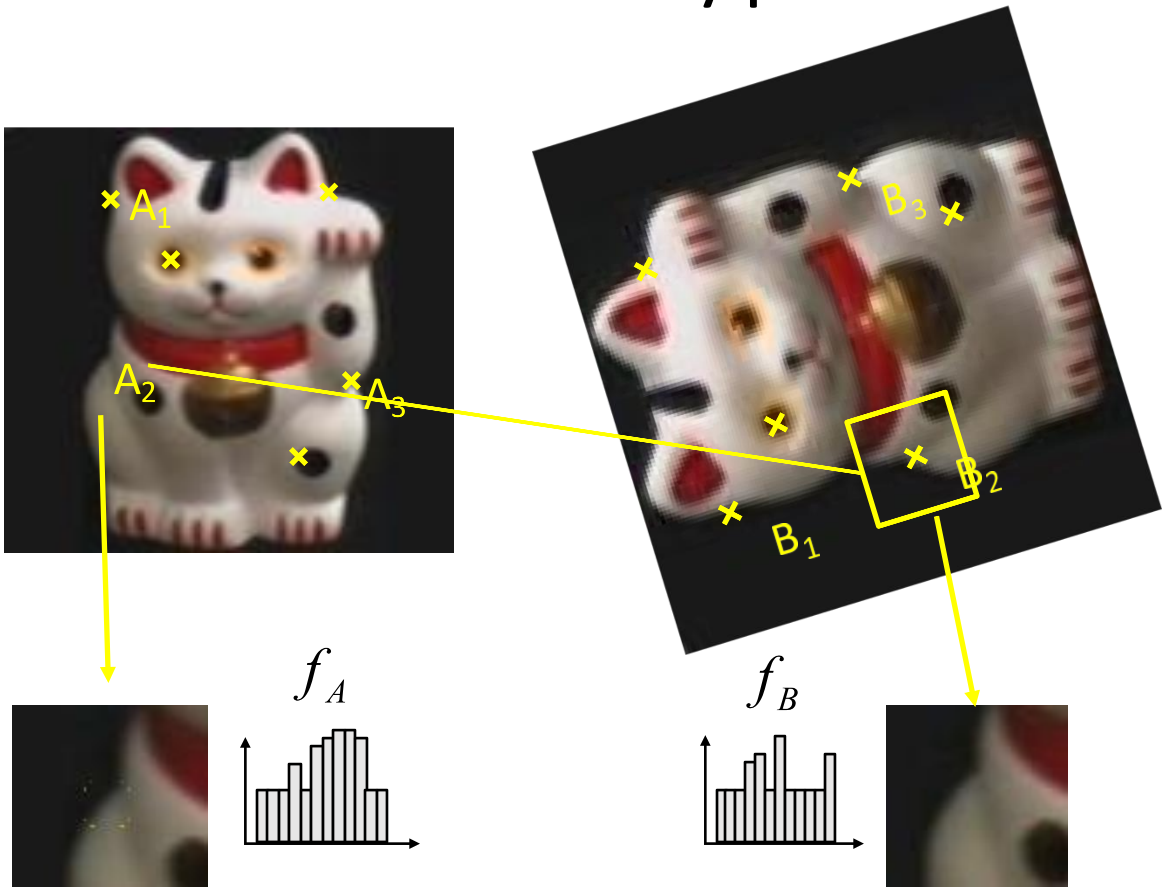
1. **Find a set of distinctive key-points**
2. **Define a region around each keypoint**

Overview of Keypoint Matching



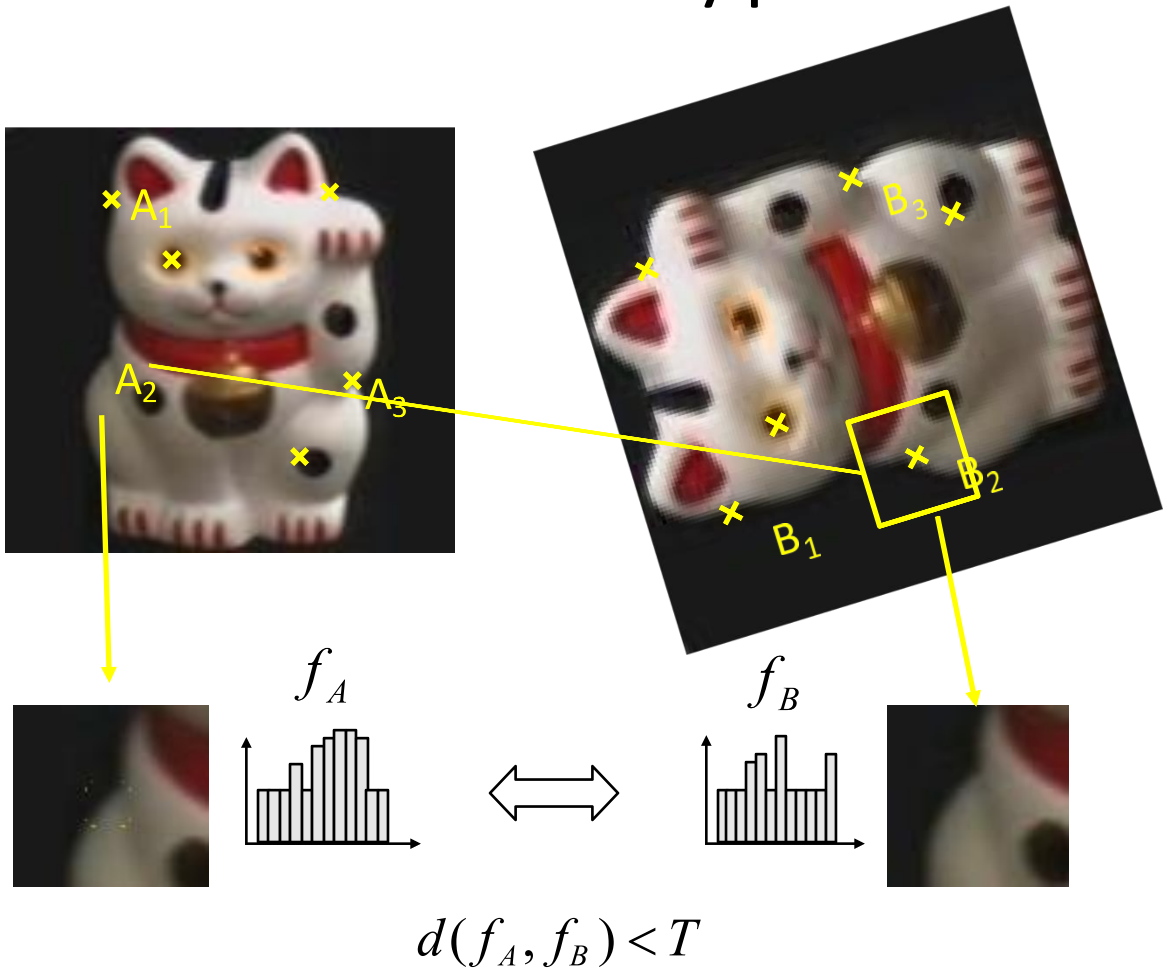
1. **Find a set of distinctive key-points**
2. **Define a region around each keypoint**
3. **Extract and normalize the region content**

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 (feature vector) from the normalized region**

Overview of Keypoint Matching

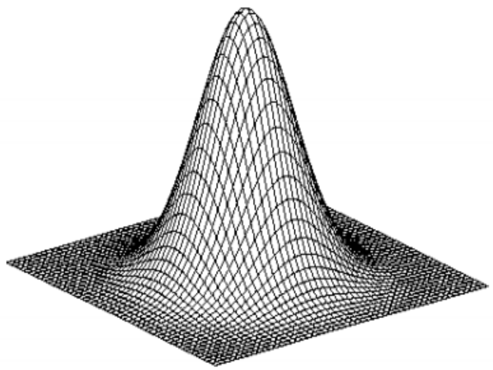


1. **Find a set of distinctive key-points**
2. **Define a region around each keypoint**
3. **Extract and normalize the region content**
4. **Compute a local descriptor (feature vector) from the normalized region**
5. **Match local descriptors**

Goals for Keypoints

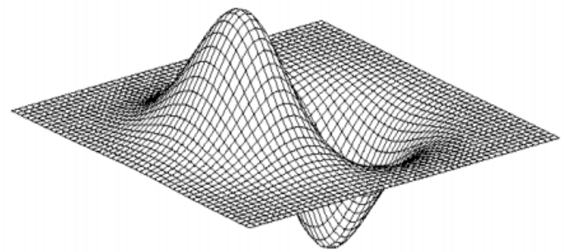
Detect points that are *repeatable* and *distinctive*

Laplacian of Gaussian



Gaussian

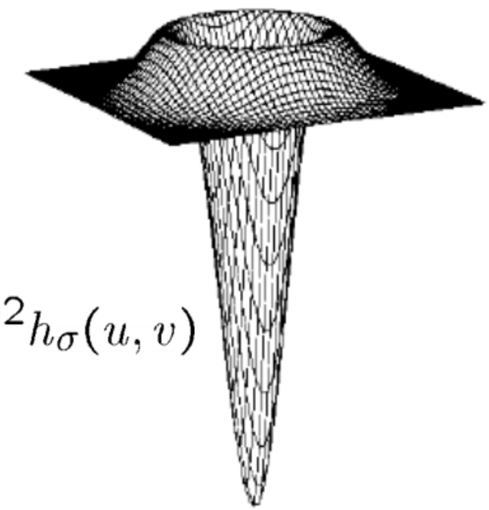
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



Derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Laplacian of Gaussian



$$\nabla^2 h_\sigma(u,v)$$

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

Laplacian of Gaussian

Example of a 3×3 Laplacian of Gaussian filter:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

How did we obtain this filter?

- Discrete approximation of the second derivative:

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

Substituting in the LoG equation: $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 y}{\partial y^2}$

$$\nabla^2 f = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

- **Converting this equation to a filter results in the given LoG matrix.**

Laplacian of Gaussian

Original Image.

Laplacian of Gaussian

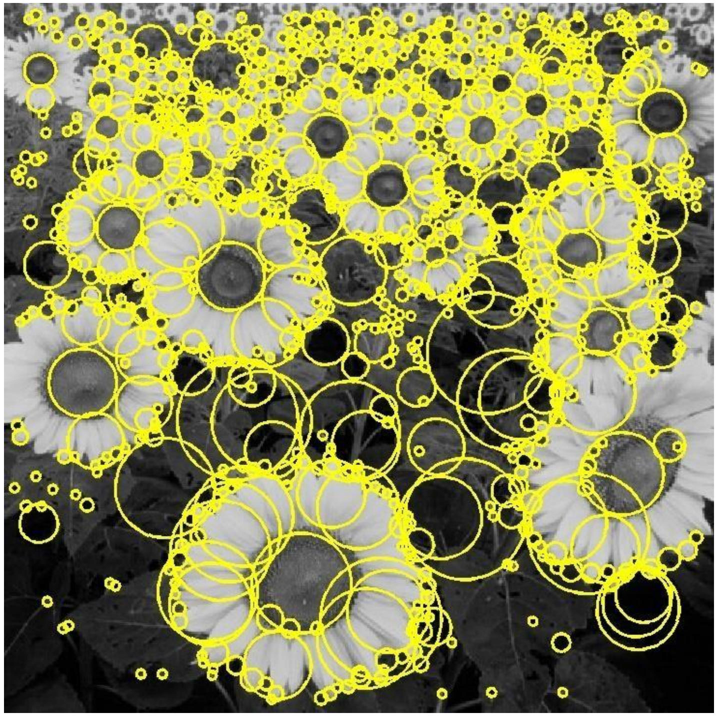
What else can LoG do? → Blob detection

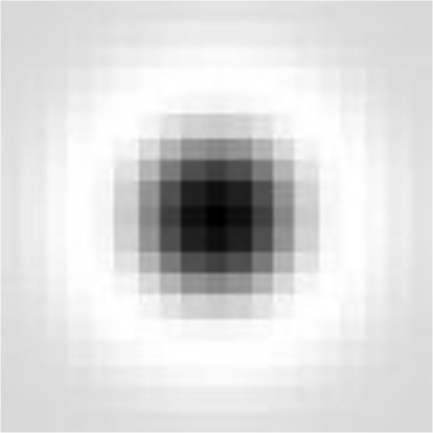
Blob detection

- Blobs: regions in an image that differ in properties (like brightness, colour, or texture) compared to surrounding areas.
- A blob is simply a group of connected pixels that share some common property (e.g., intensity > threshold).
- A *blob* could be circular, oval, or just any connected region of pixels.
- Blobs often correspond to objects or features in the image.

Basic idea

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

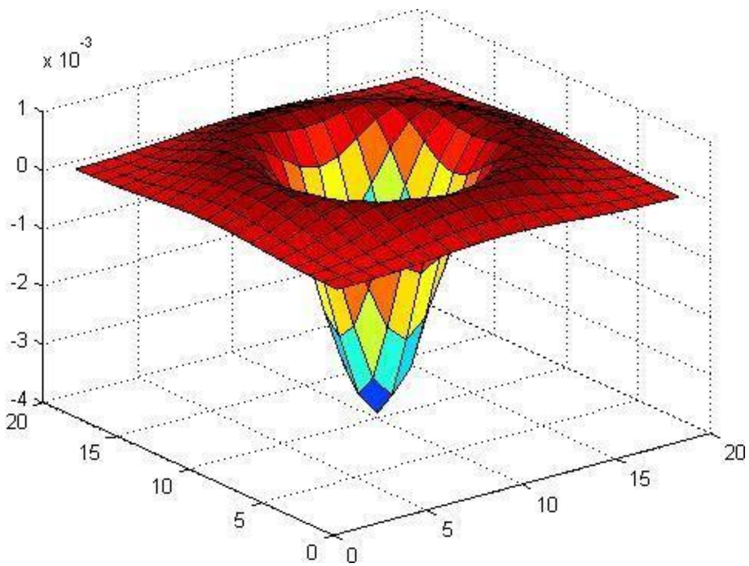


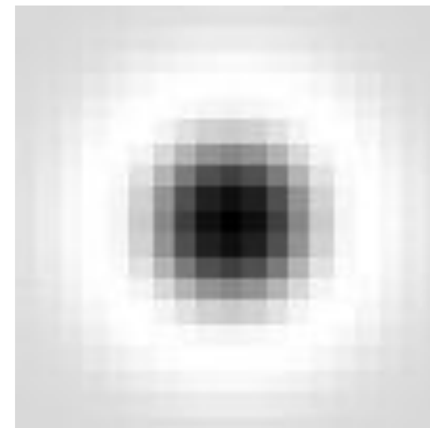


T. Lindeberg. [Feature detection with automatic scale selection.](#) *IJCV* 30(2), pp 77-116, 1998.

LoG as Blob filter

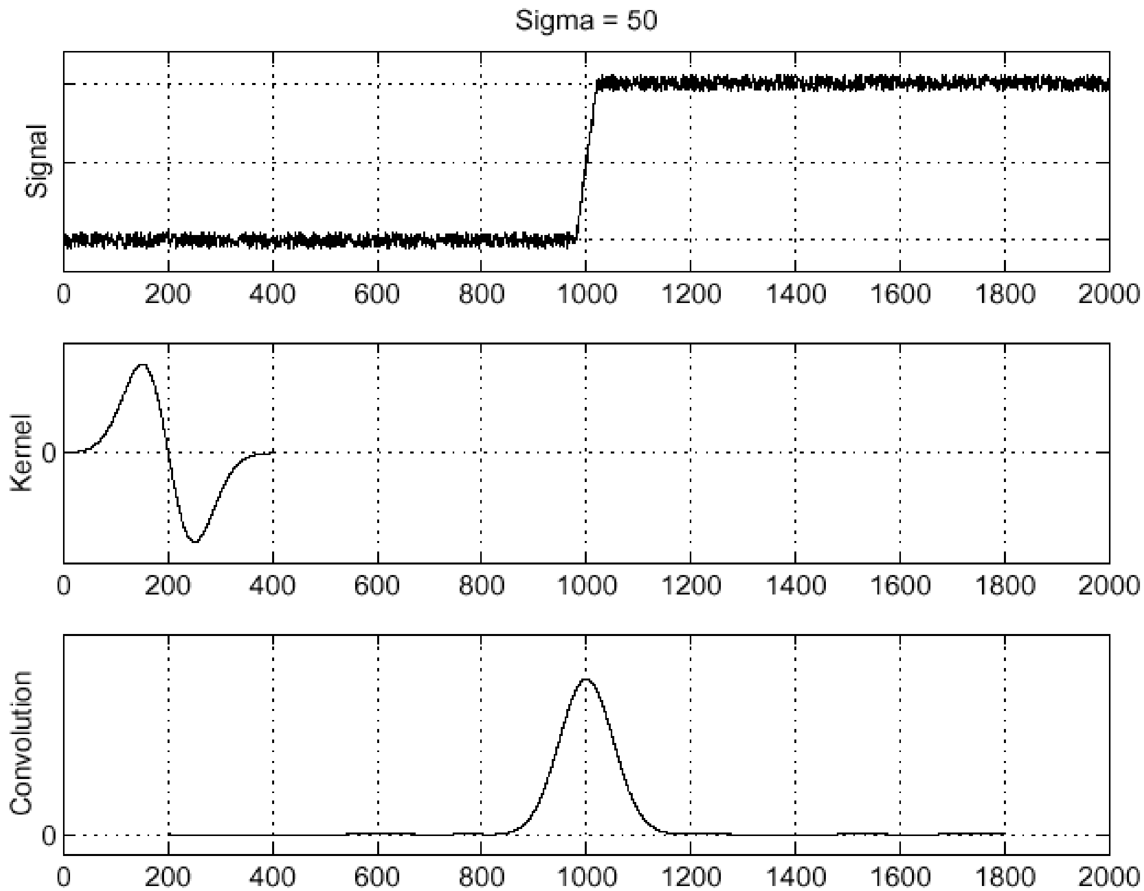
- Laplacian of Gaussian: Circularly symmetric operator, can be used for blob detection in 2D





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

Recall: Edge detection



f — Edge

$\frac{d}{dx}g$ — Derivative of Gaussian

$f * \frac{d}{dx}g$ — Edge = maximum of derivative

Source: S. Seitz

Edge detection, Take 2

f

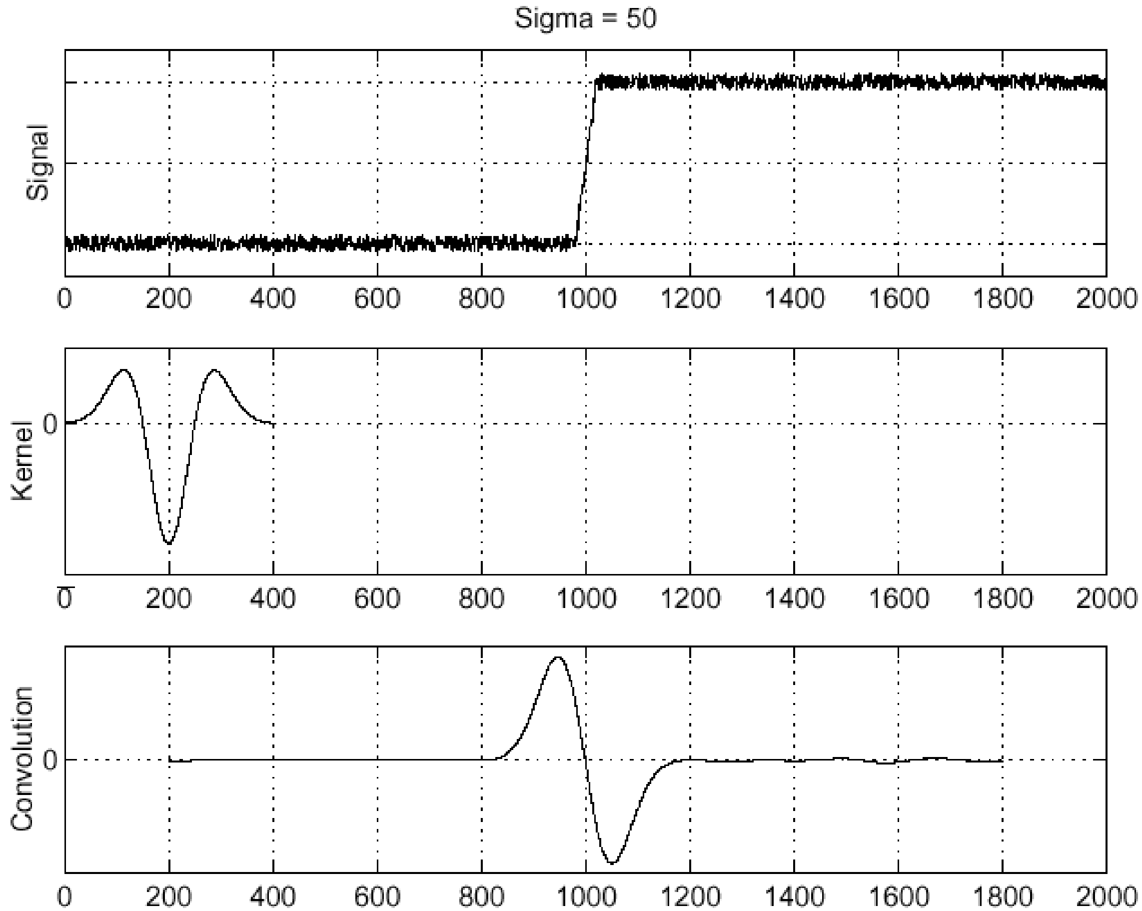
Edge

$$\frac{d^2}{dx^2} g$$

Second derivative of Gaussian (Laplacian)

$$f * \frac{d^2}{dx^2} g$$

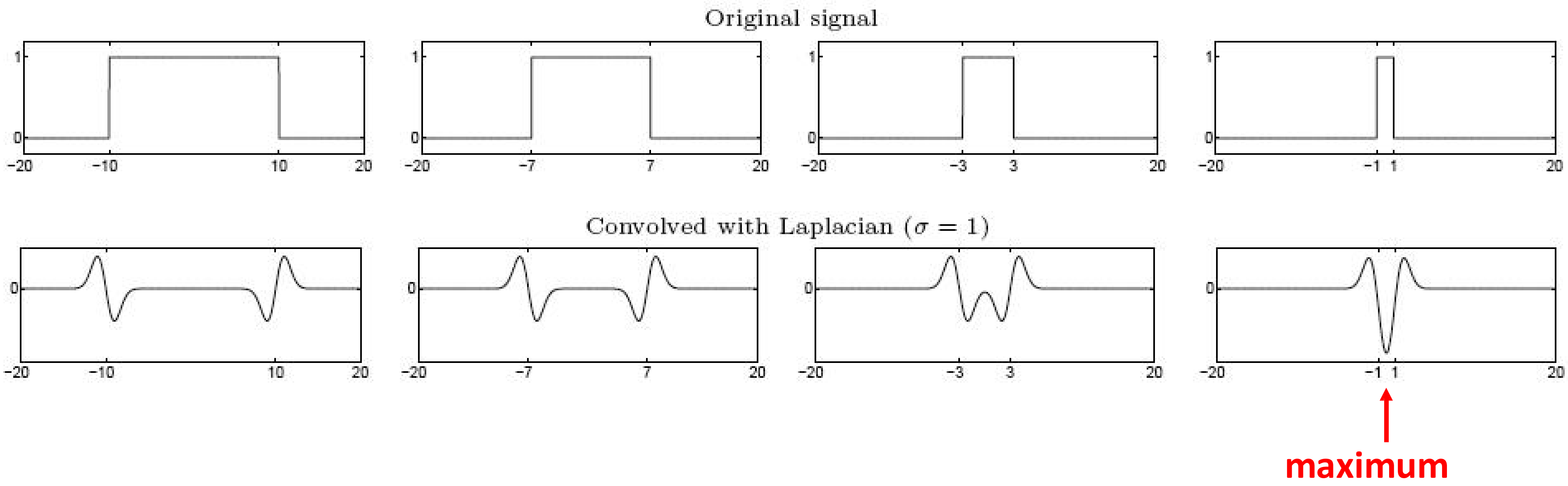
Edge = zero crossing of second derivative



Source: S. Seitz

From edges to blobs

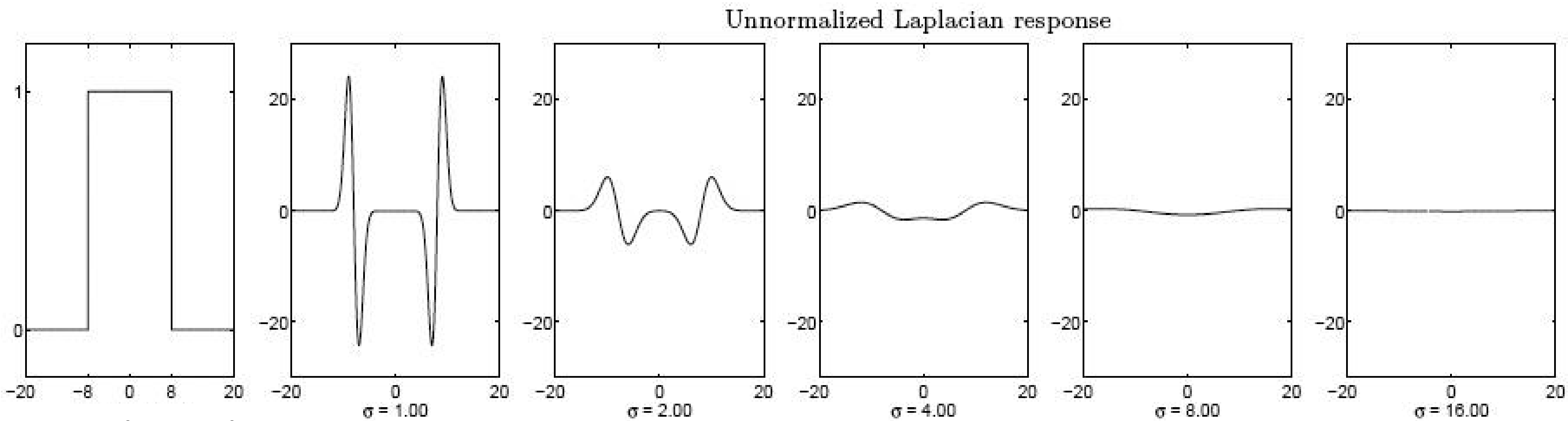
- Edge = ripple
- Blob = superposition of two ripples



Spatial selection: the magnitude of the Laplacian response will achieve a maximum at the center of the blob, provided the scale of the Laplacian is "matched" to the scale of the blob

Scale selection

- The scale of the blob is found by convolving it with Laplacians at several scales and looking for the maximum response
- However, Laplacian response decays as scale increases:



original signal
(radius=8)

increasing σ

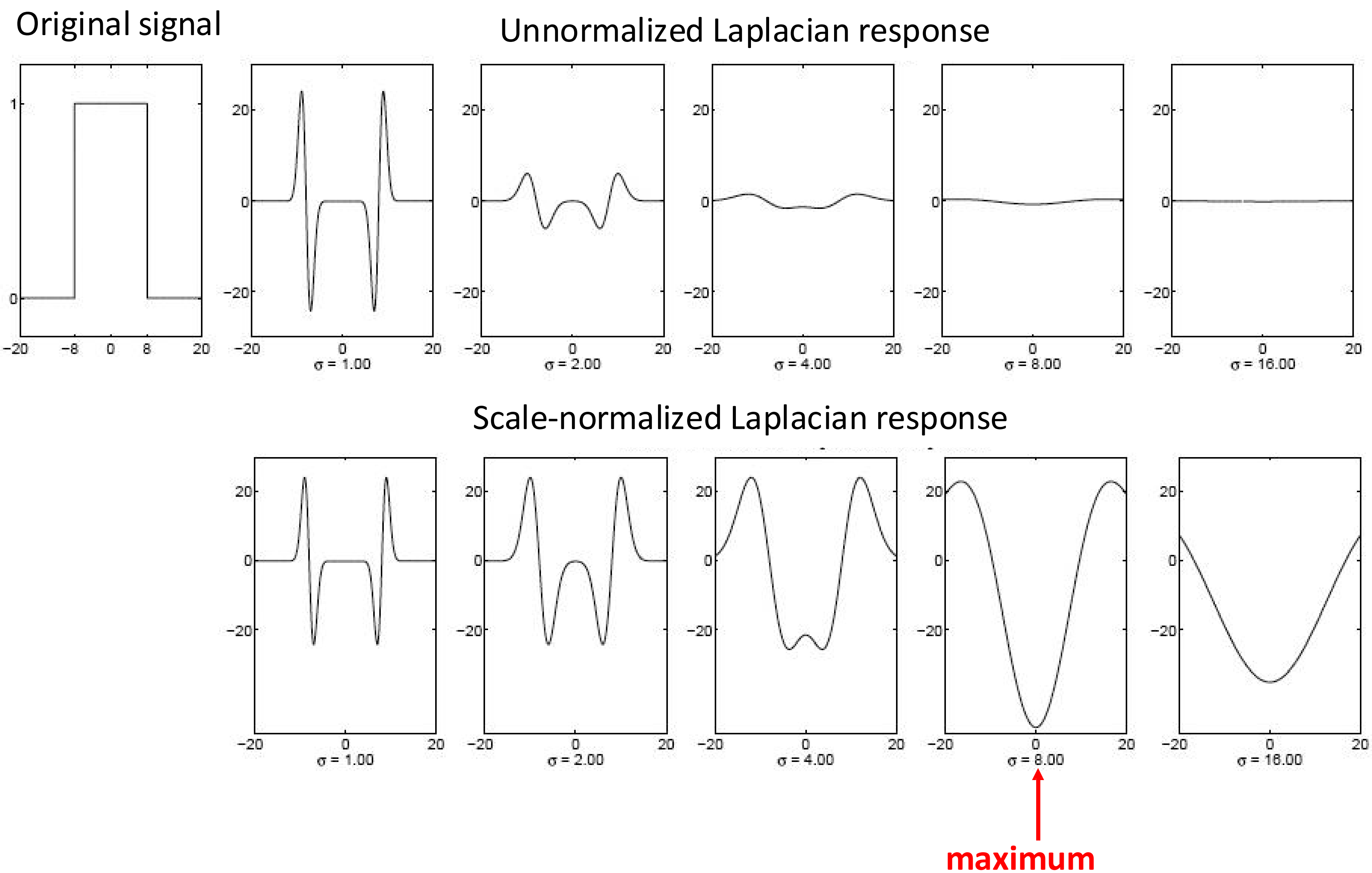
Scale normalization

- The response of a derivative of Gaussian filter to a perfect step edge decreases as σ increases

- To keep response the same (scale-invariant), must multiply Gaussian derivative by σ

- Laplacian is the second Gaussian derivative, so it must be multiplied by σ^2

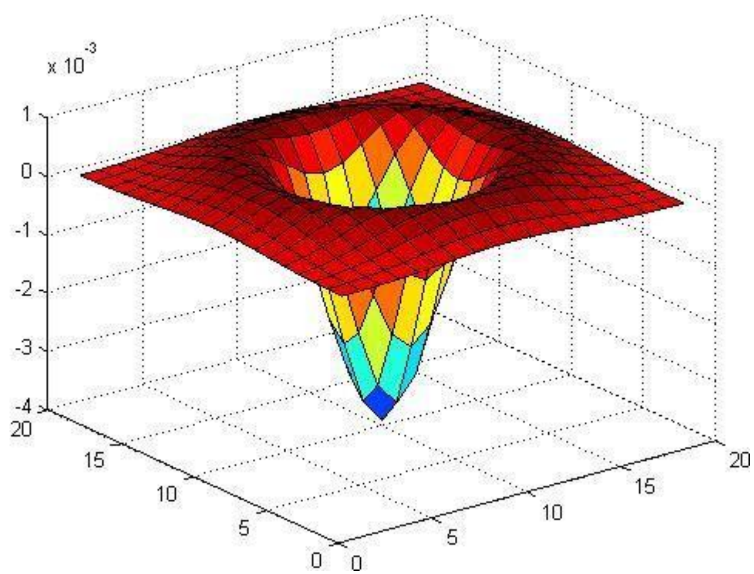
Effect of scale normalization

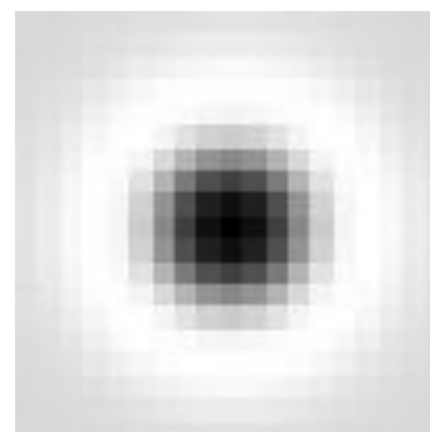


Blob detection in 2D

- *Scale-normalized* Laplacian of Gaussian:

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



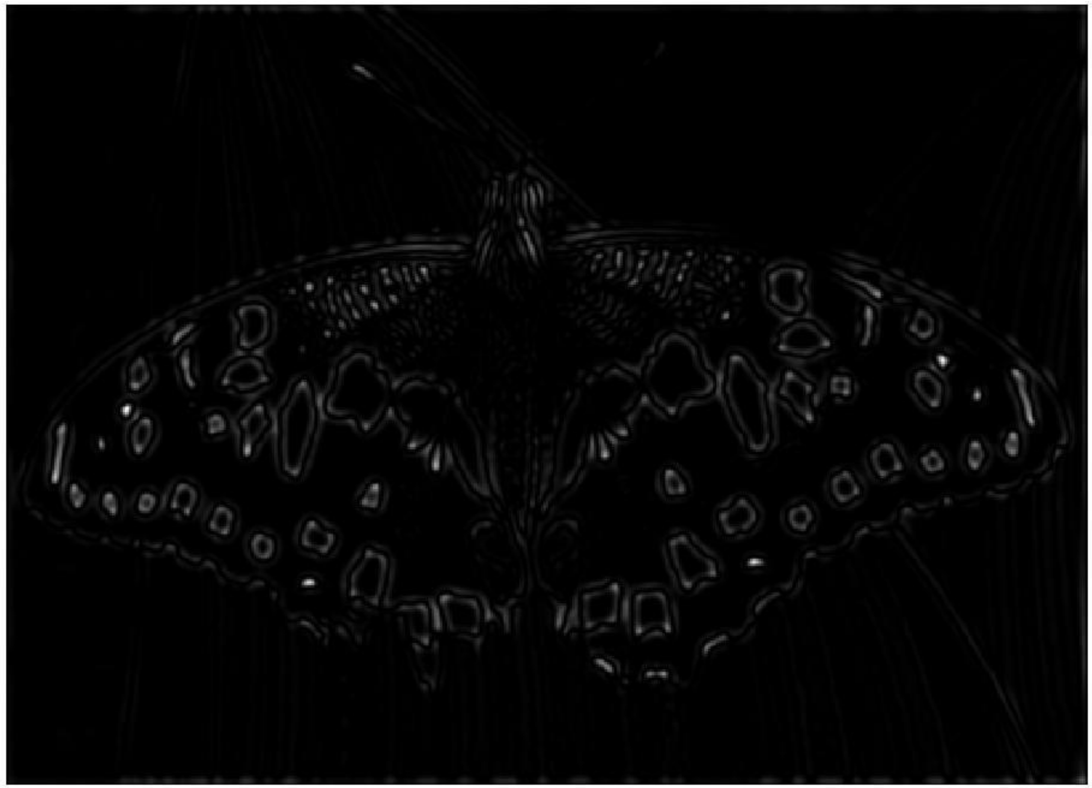


Scale-space blob detector

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

Scale-space blob detector: Example

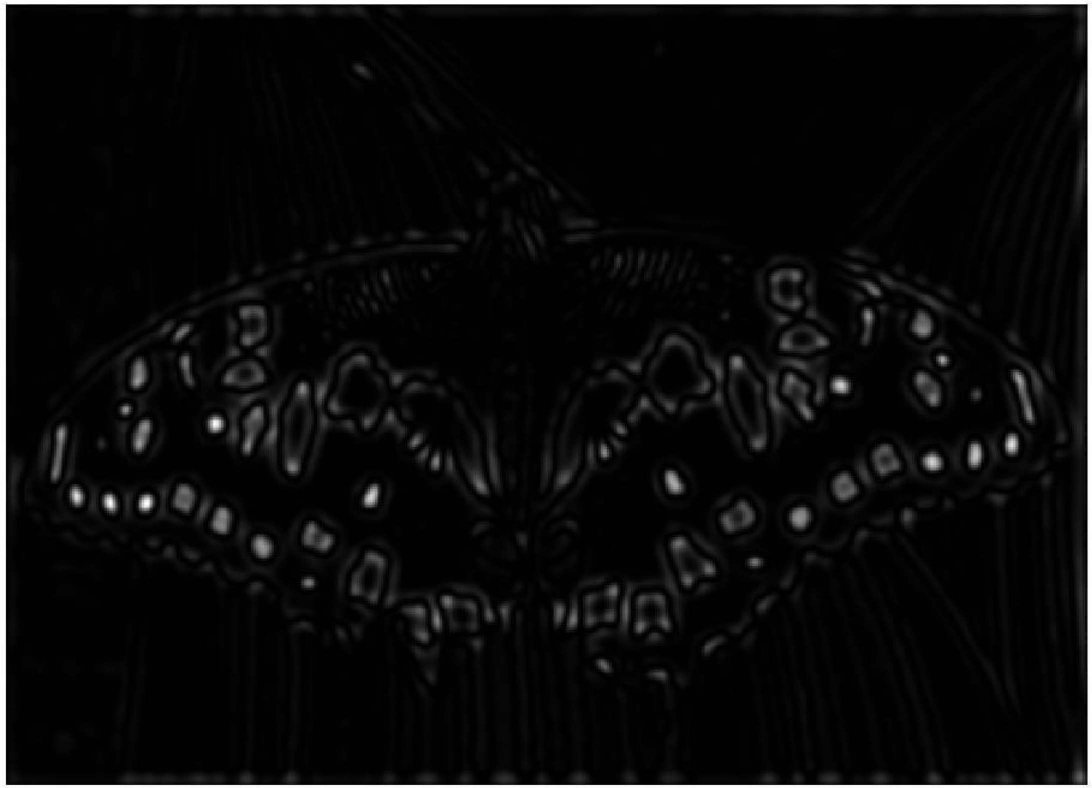


sigma = 2

Scale-space blob detector: Example

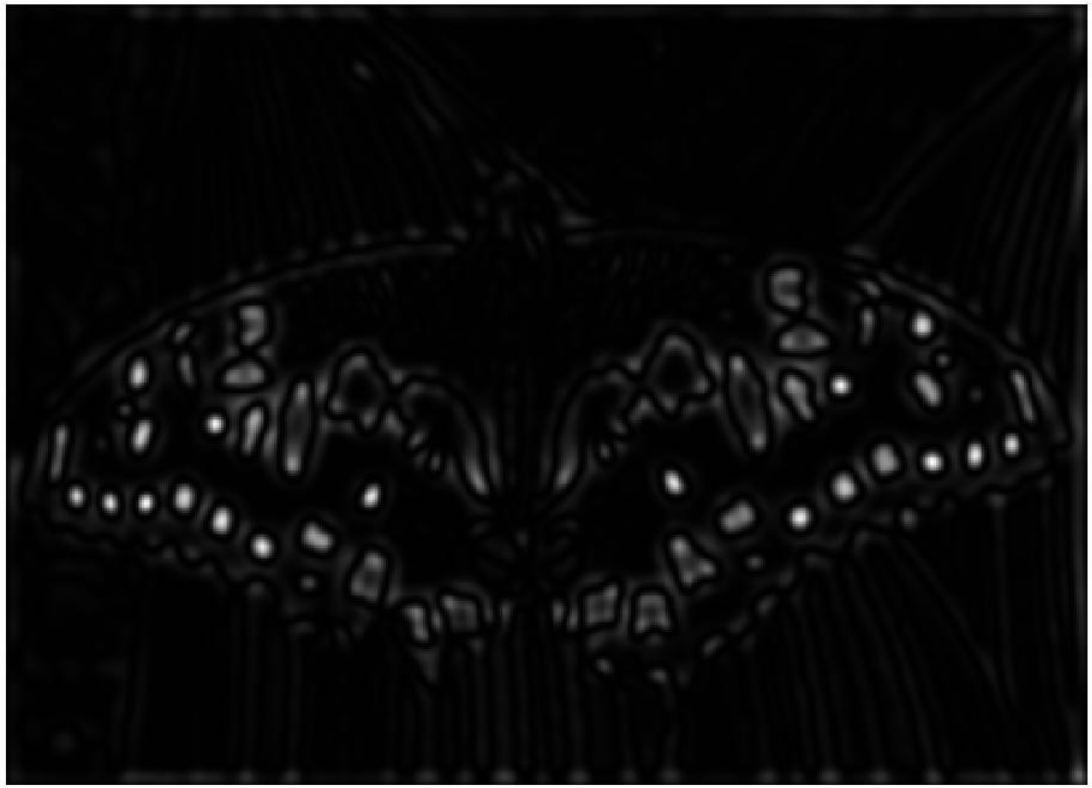
sigma = 2.5018

Scale-space blob detector: Example



sigma = 3.1296

Scale-space blob detector: Example

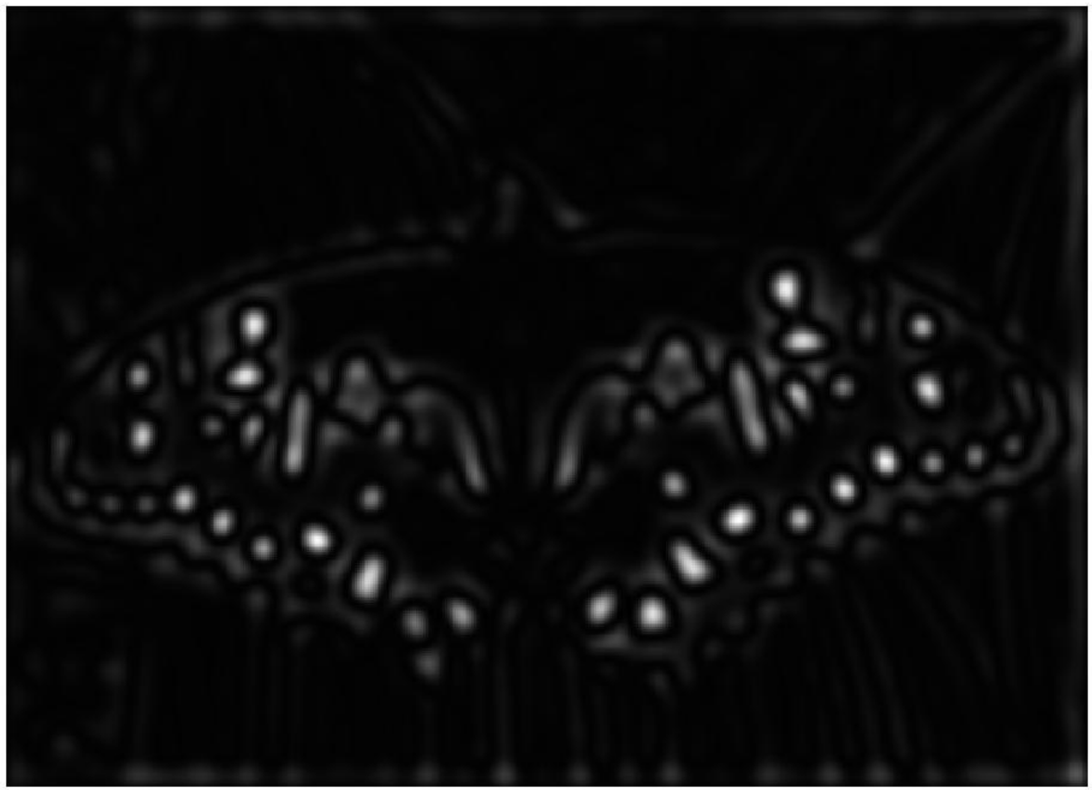


sigma = 3.9149

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

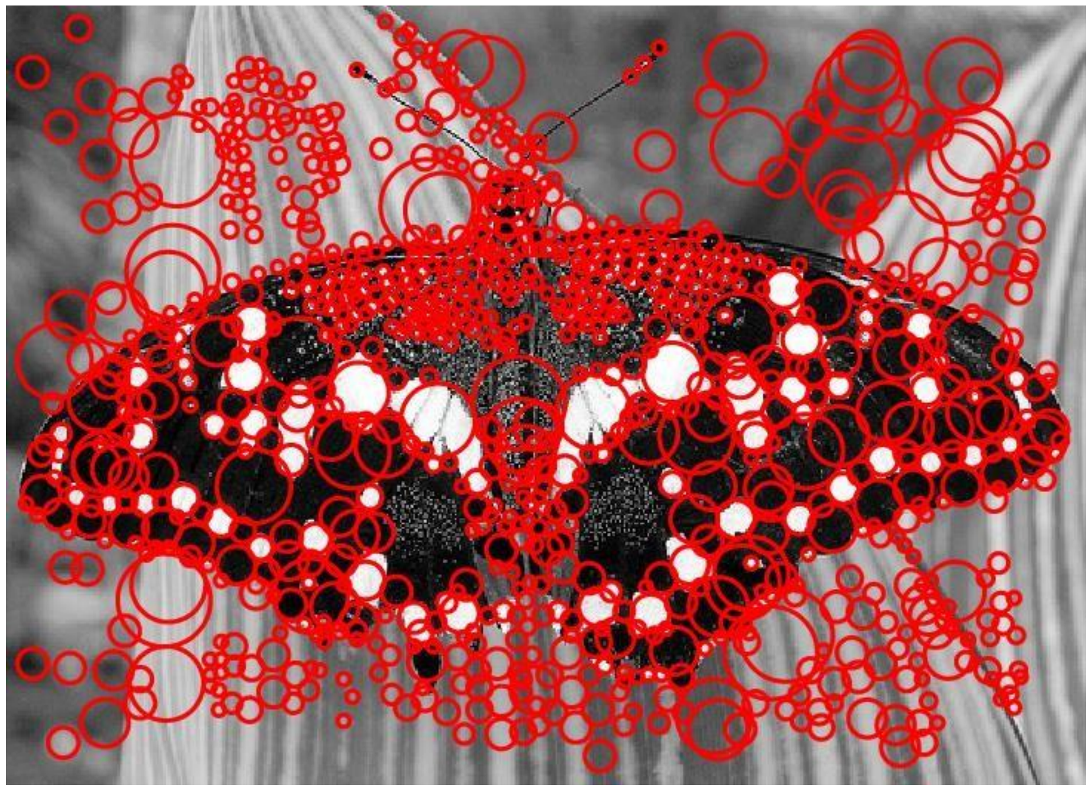
Scale-space blob detector: Example

sigma = 11.9912

Scale-space blob detector: Example

sigma = 15

Scale-space blob detector: Example



From Blobs to Corners

- In the following image, what are some interesting features to choose?

Credit: *K Grauman, R Urtasun*

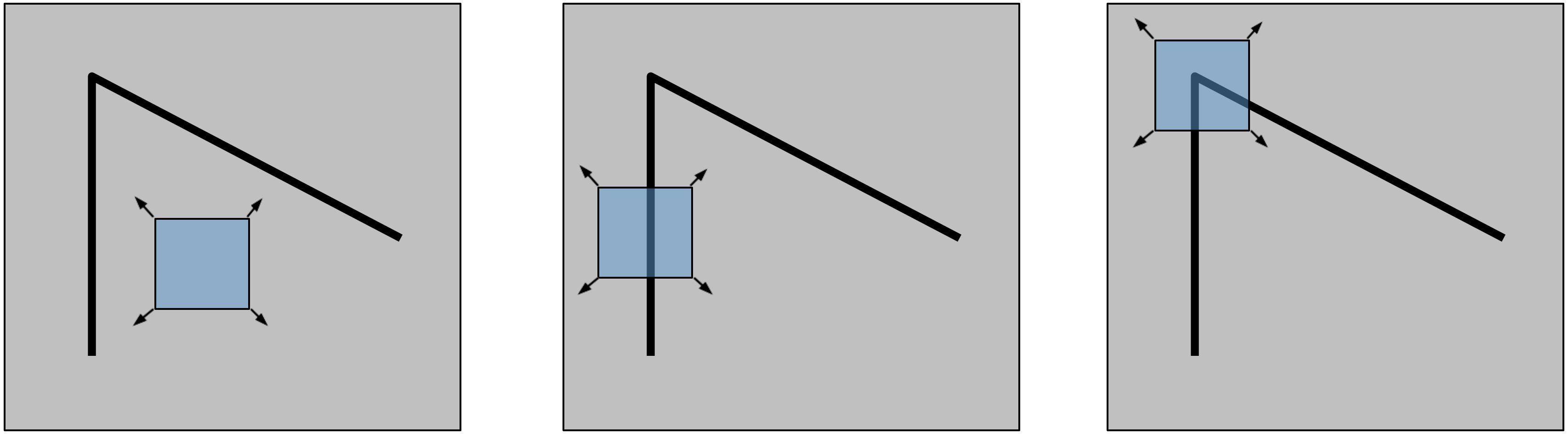
From Blobs to Cornes

- Look for image regions that are unusual. How to define "unusual"?
- Texture-less patches are nearly impossible to localize.
- Patches with large contrast changes (gradients) are easier to localize.
- But straight line segments at a single orientation suffer from the aperture problem (we'll see next slide), i.e., it is only possible to align the patches along the direction normal to the edge direction.
- Gradients in at least two (significantly) different orientations are the easiest, e.g., corners.

Credit: *K Grauman, R Urtasun*

Corner Detection: Basic Idea

- Consider a small window of pixels.
- How does the window change when you shift it?



"flat" region:
no change in all directions

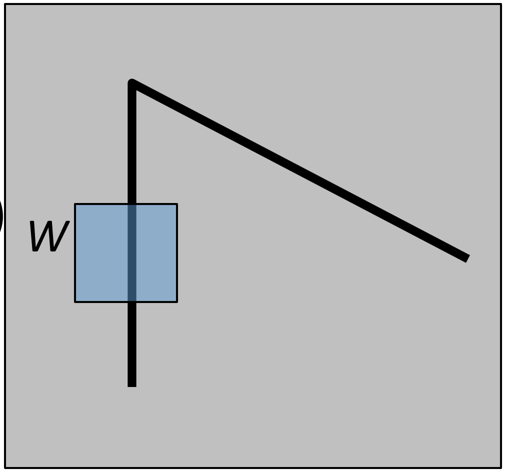
"edge":
no change along the edge direction

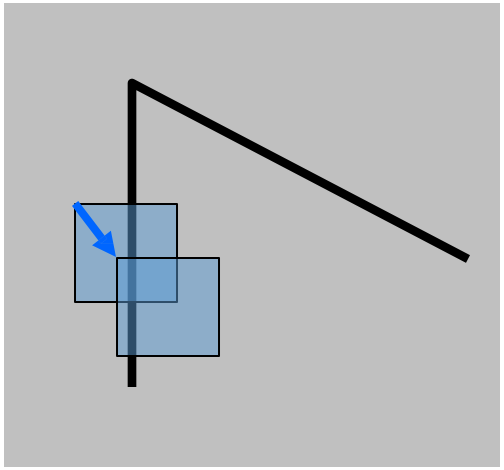
"corner":
significant change in all directions

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

Computing Autocorrelation

- In the previous slide, how to quantify the "significant" change of the window?
- Answer: **Autocorrelation function** (ACF) or Sum of squared differences (SSD).

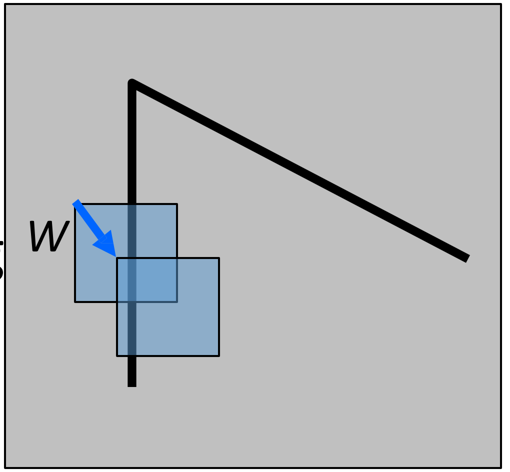




Computing Autocorrelation

Consider shifting the window W by (u,v) = $(\Delta u, \Delta v)$

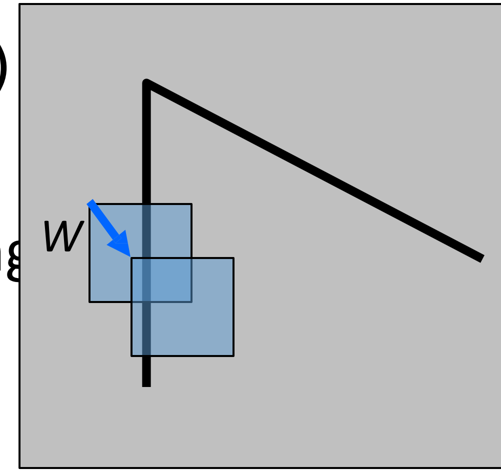
- how do the pixels in W change?
- compare each pixel before and after by finding the ACF
- this defines an ACF "error" $E(u,v)$:



Computing Autocorrelation

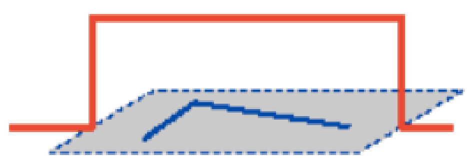
Consider shifting the window *W* by (*u,v*) = (Δu, Δv)

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



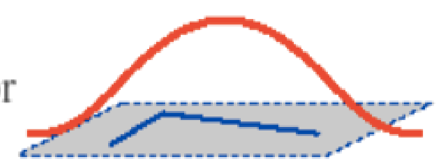
$$E(u,v) = \sum_{(x,y)\in W} w(x,y)\left(I(x+u,y+v)-I(x,y)\right)^2$$

Window function $w(x,y)$ =



1 in window, 0 outside

or



Gaussian

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*:

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

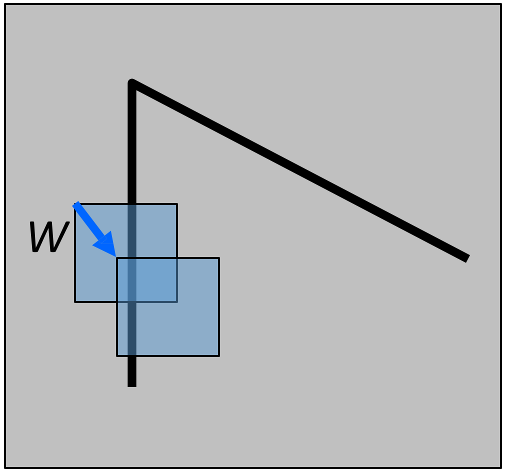
If the motion (u,v) is small, then first order approximation is good

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

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

Plugging this into the formula on the previous slide…

Corner detection: the math



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

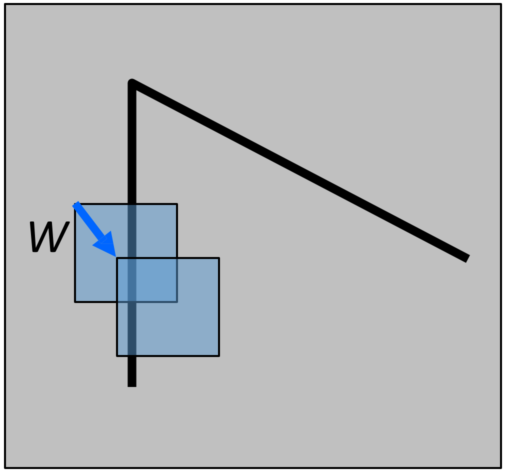
(Shorthand: $I_x = \frac{\partial I}{\partial x}$)

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

Computing Autocorrelation

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

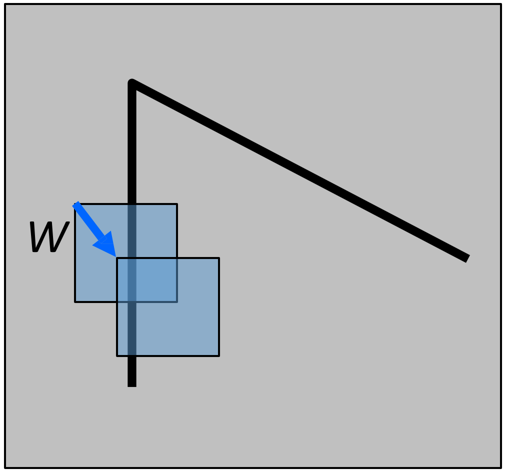
(Shorthand: $I_x = \frac{\partial I}{\partial x}$)



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

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

Computing Autocorrelation



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

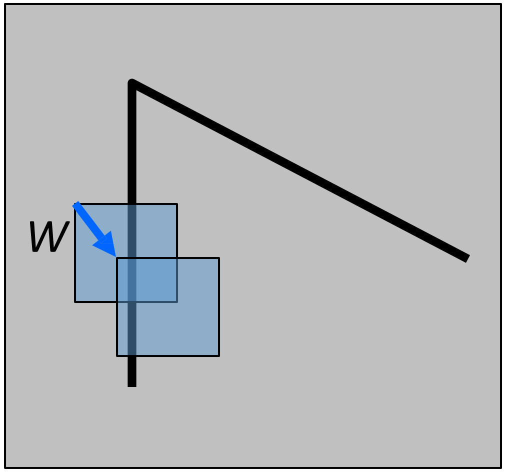
(Shorthand: $I_x = \frac{\partial I}{\partial x}$)

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

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

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

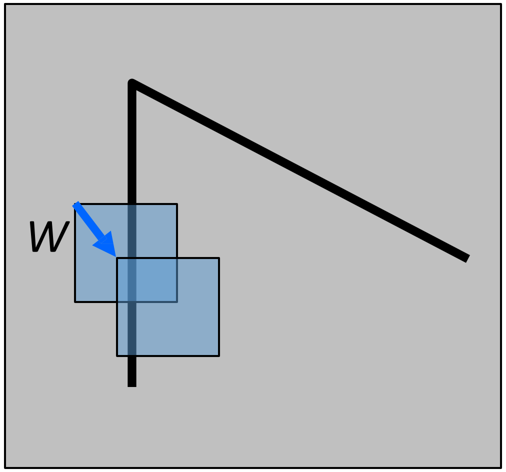
Computing Autocorrelation



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

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

Computing Autocorrelation



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

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

$$E(u,v) = [u \;\; v]\; H \;[u \;\; v]^T$$

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

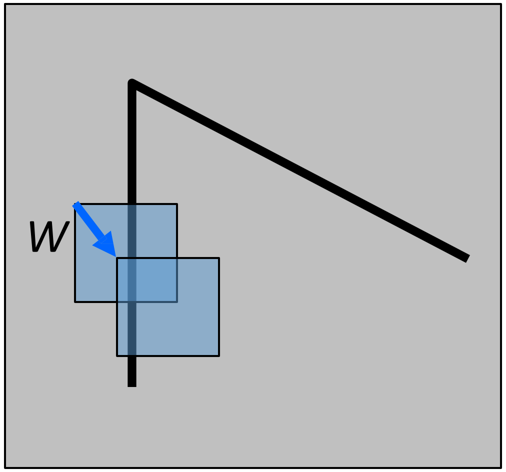
- Thus, $E(u,v)$ is locally approximated as a *quadratic form*

Computing Autocorrelation

- The weighted summations have been replaced with discrete convolutions with the weighting kernel w.
- The eigenvalues of **H** reveal the amount of intensity change in the two principal orthogonal gradient directions in the window.

$$H = \mathbf{U}\begin{bmatrix}\lambda_1 & 0 \\ 0 & \lambda_2\end{bmatrix}\mathbf{U}^T \qquad \text{with} \qquad H\mathbf{u}_i = \lambda\mathbf{u}_i$$

Computing Autocorrelation



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

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

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

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

- 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.

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

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

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

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

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.

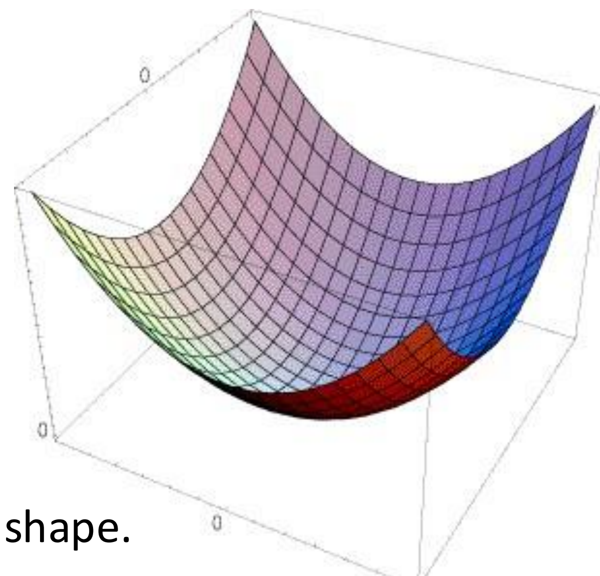
$$E(u,v) \approx Au^2 + 2Buv + Cv^2$$

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

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

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

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



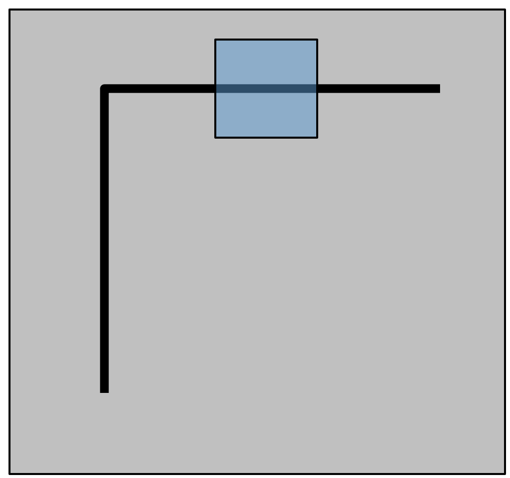
Let’s try to understand its shape.

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

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

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

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



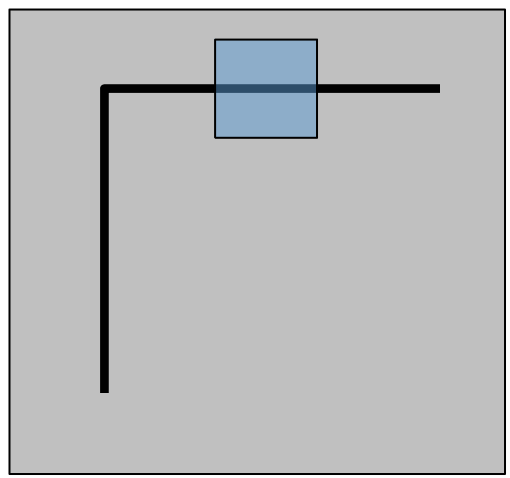
Horizontal edge: $I_x = 0$

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

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

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

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



Horizontal edge: $I_x = 0$

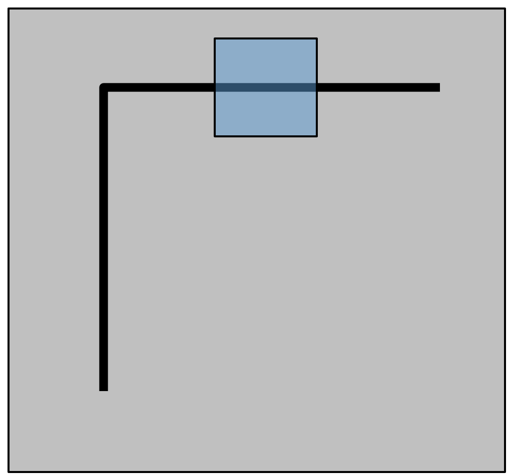
$$H = \begin{bmatrix} 0 & 0 \\ 0 & C \end{bmatrix}$$

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

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

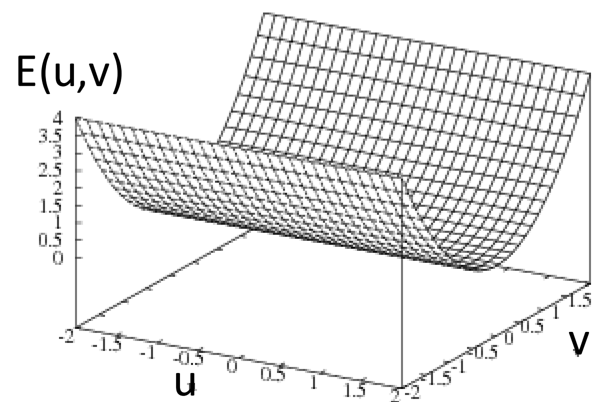
$$B = \sum_{(x,y)\in W} I_x I_y$$

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



Horizontal edge: $I_x = 0$

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

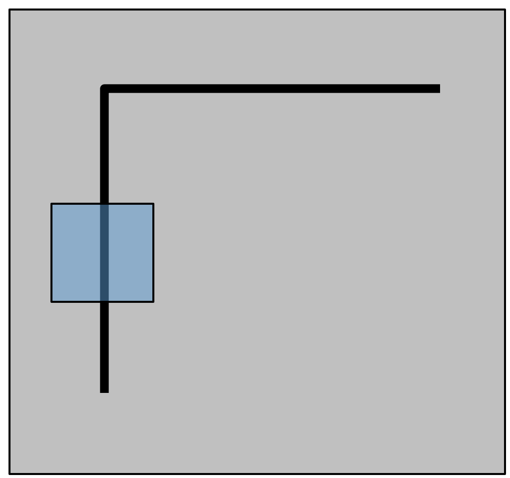


$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

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

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

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



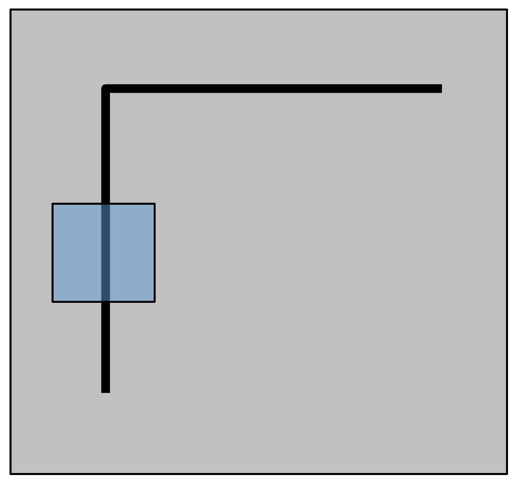
Vertical edge: $I_y = 0$

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

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

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

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



Vertical edge: $I_y = 0$

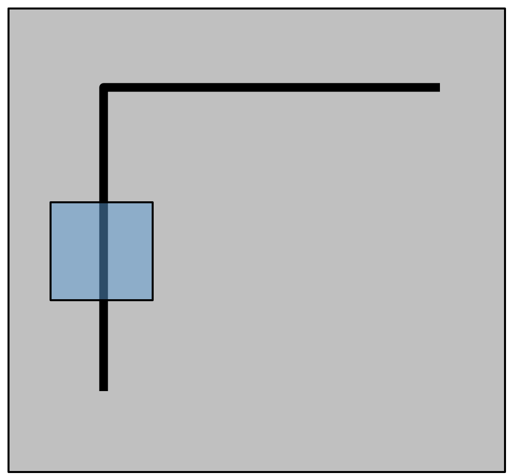
$$H = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}$$

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

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

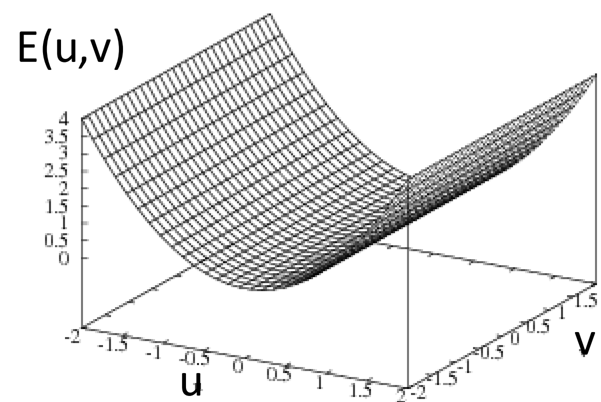
$$B = \sum_{(x,y)\in W} I_x I_y$$

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



Vertical edge: $I_y = 0$

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

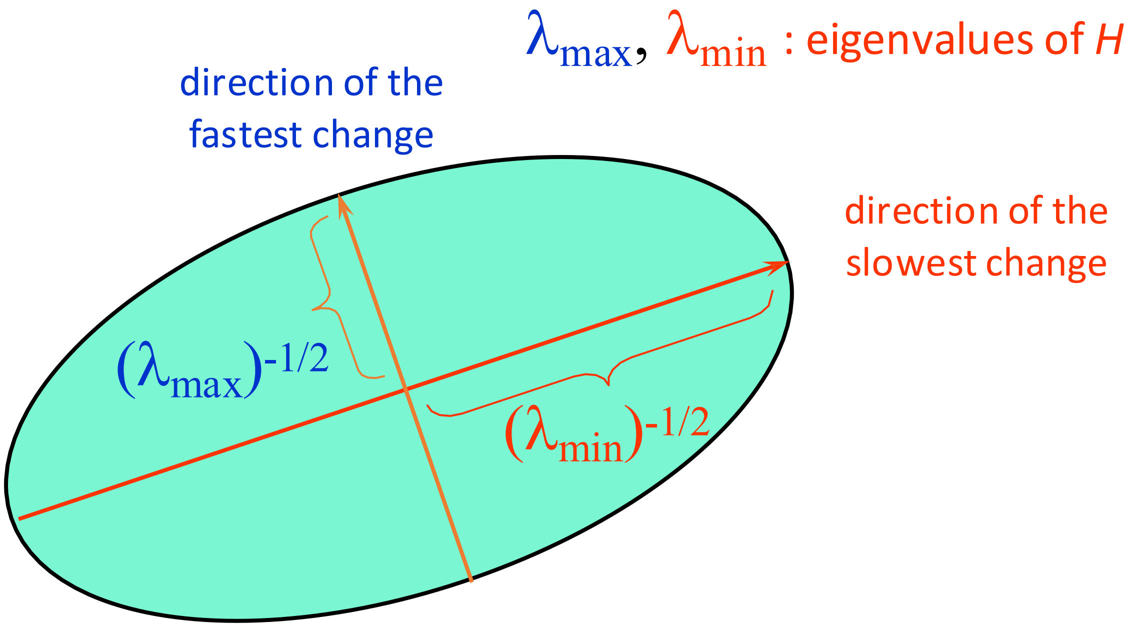


Interpretation

- 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:

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

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

$$det \left[\begin{array}{cc} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{array} \right] = 0$$

- The solution:

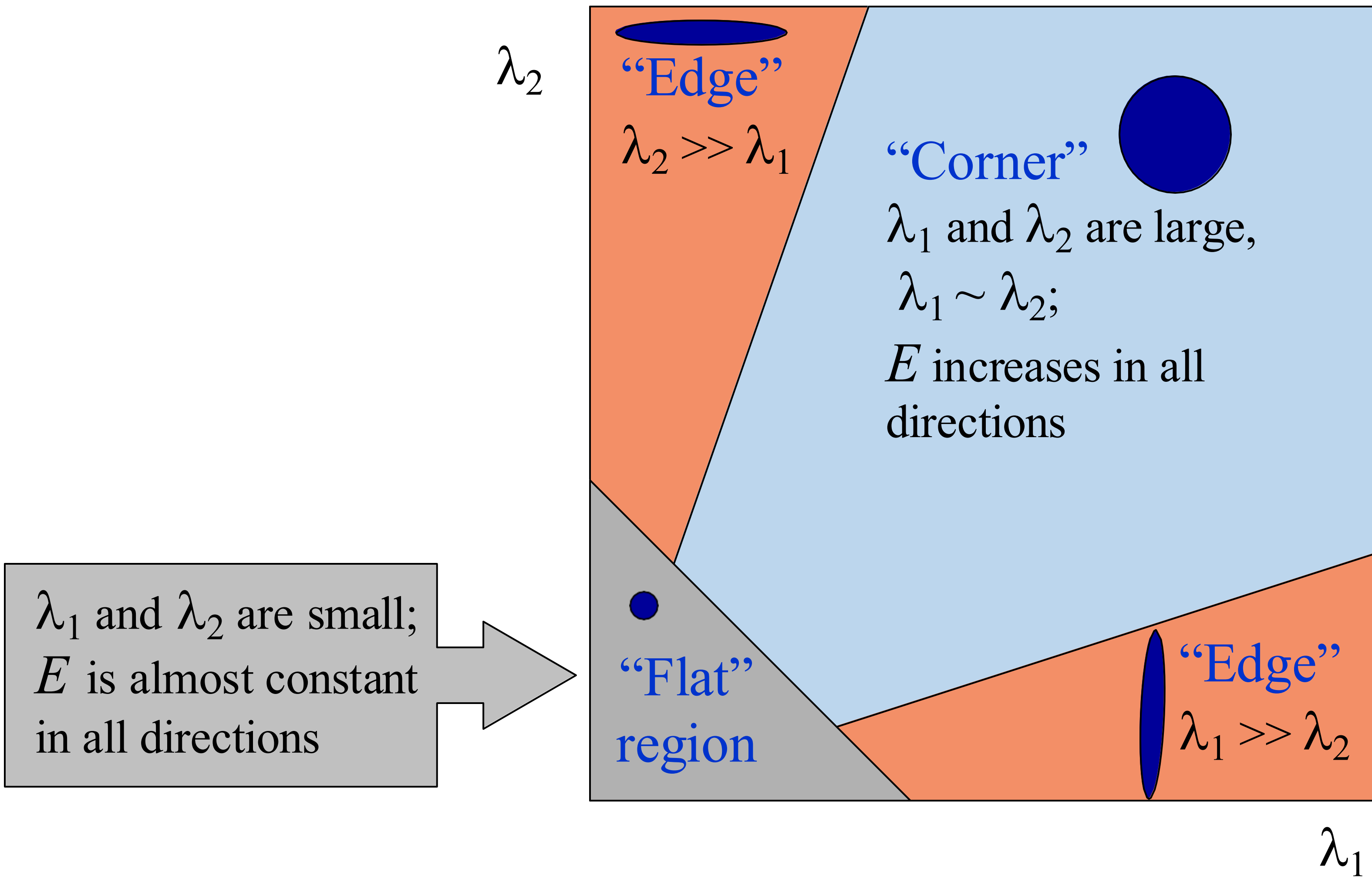
$$\lambda_{\pm} = \frac{1}{2}\left[(h_{11} + h_{22}) \pm \sqrt{4h_{12}h_{21} + (h_{11} - h_{22})^2} \right]$$

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

$$\left[\begin{array}{cc} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{array} \right] \left[\begin{array}{c} x \\ y \end{array} \right] = 0$$

Interpreting the eigenvalues

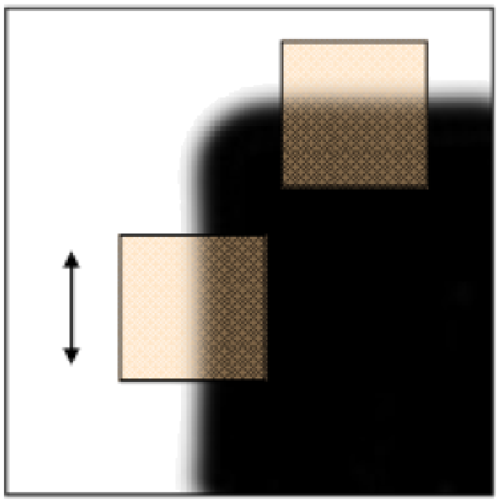
How do the eigenvalues determine if an image point is a corner?



Credit: N Snavely, R Urtasun

Interpreting the eigenvalues

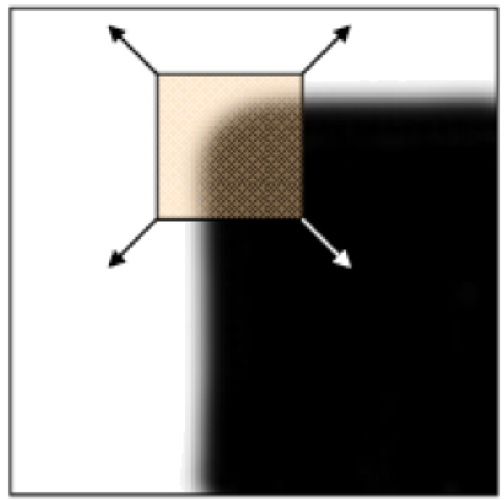
How do the eigenvalues determine if an image point is a corner?



"edge":

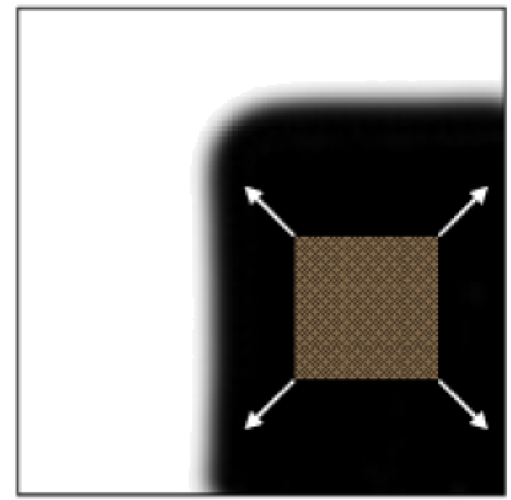
$\lambda_1 >> \lambda_2$

$\lambda_2 >> \lambda_1$



"corner":

λ_1 and λ_2 are large,
$\lambda_1 \sim \lambda_2$;



"flat" region

λ_1 and λ_2 are small;

Credit: N Snavely, R Urtasun

Harris Corner Detector

Here’s what you do

- Compute the gradient at each point in the image
- Compute **H** for each image window to get its cornerness scores.
- Compute the eigen values or compute the following function

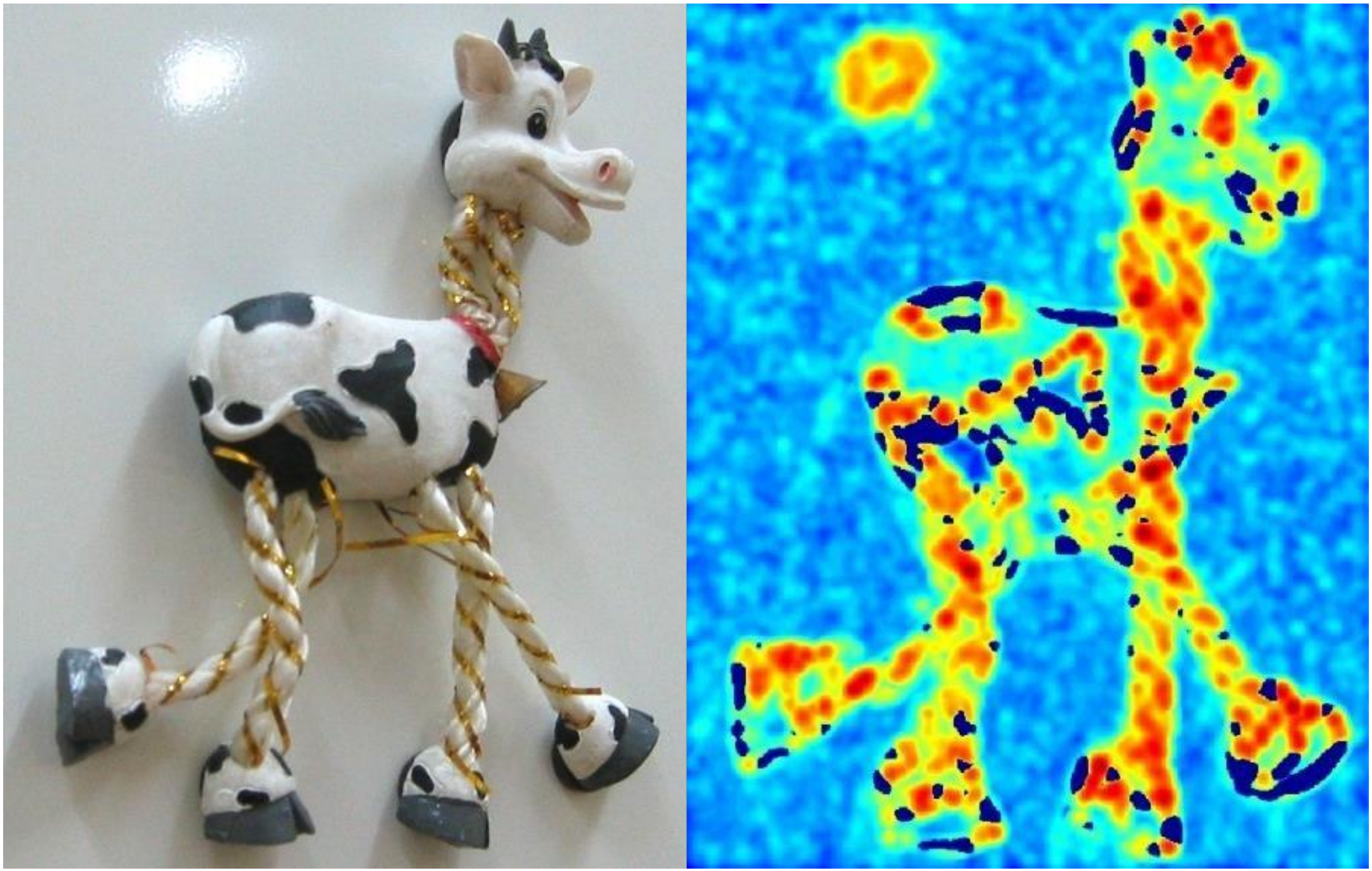
Mc = $\lambda_1 \lambda_2 - \kappa \, (\lambda_1 + \lambda_2)^2$ = det(H) - k trace^2(H)

- Find points whose surrounding window gave larger cornerness response (Mc > threshold)
- Take points of local maxima, perform non-maximum suppression.

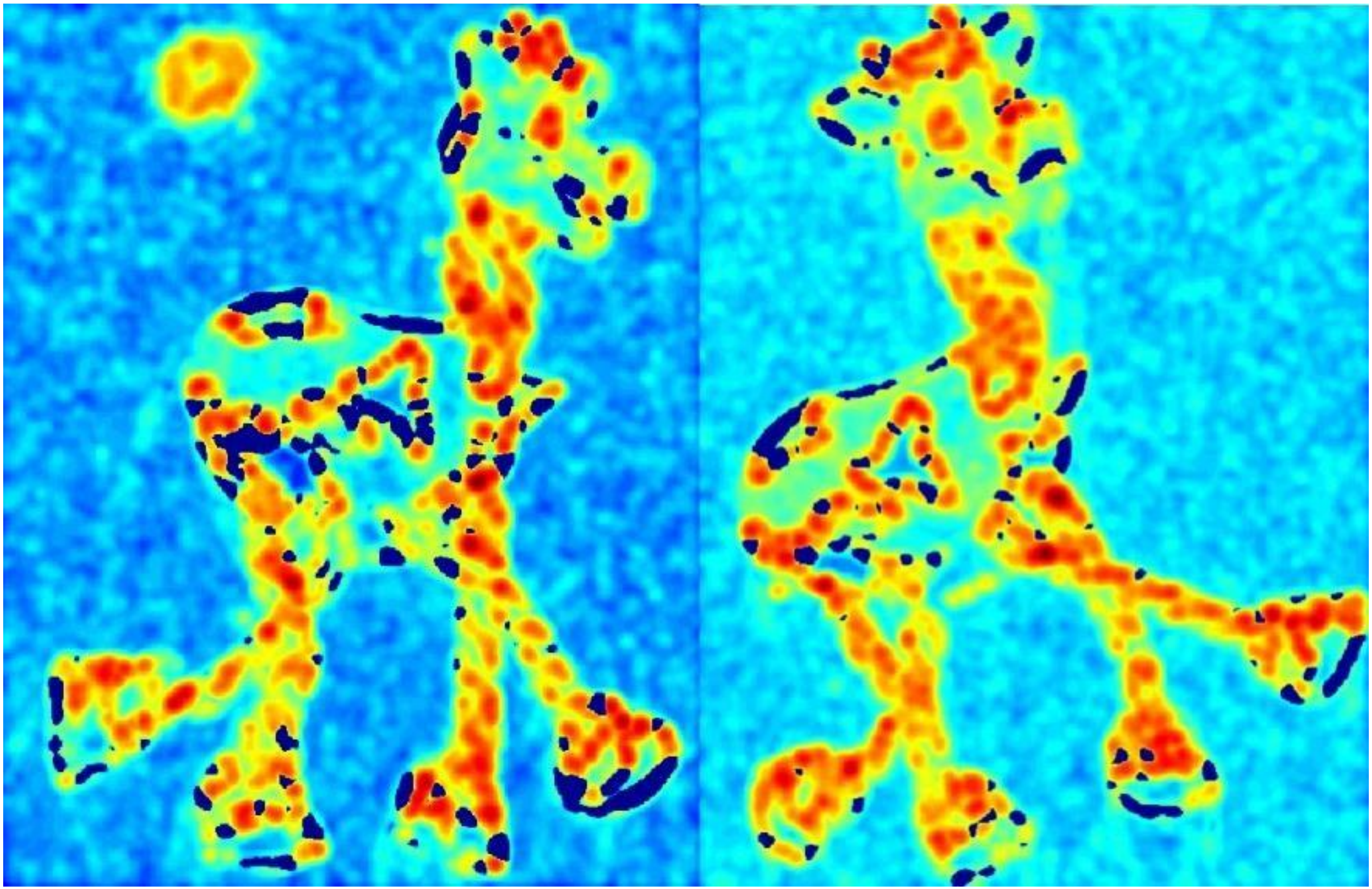
Credit: N Snavely, R Urtasun

Harris detector example

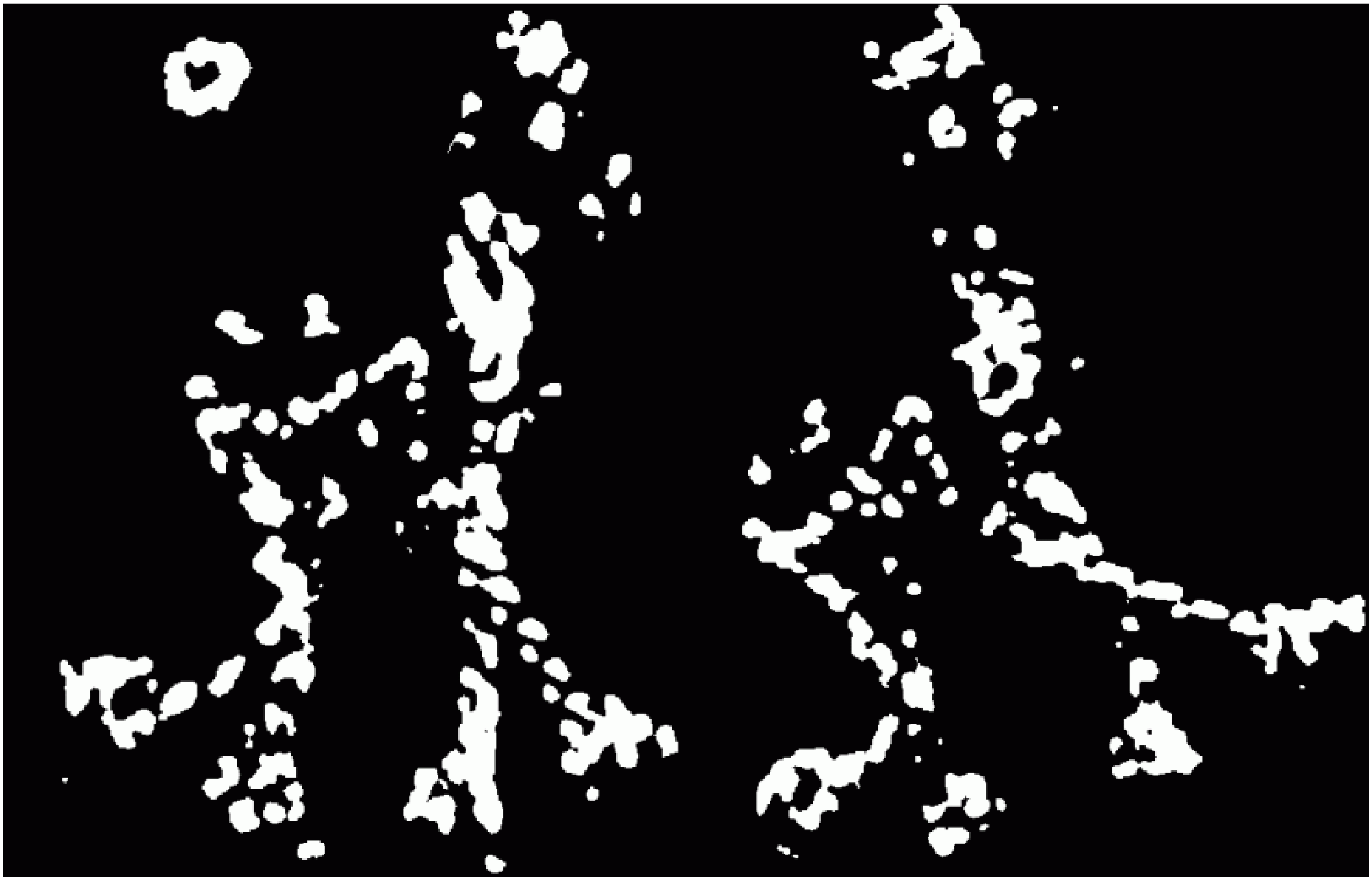
Computer cornerness scores (red high, blue low)



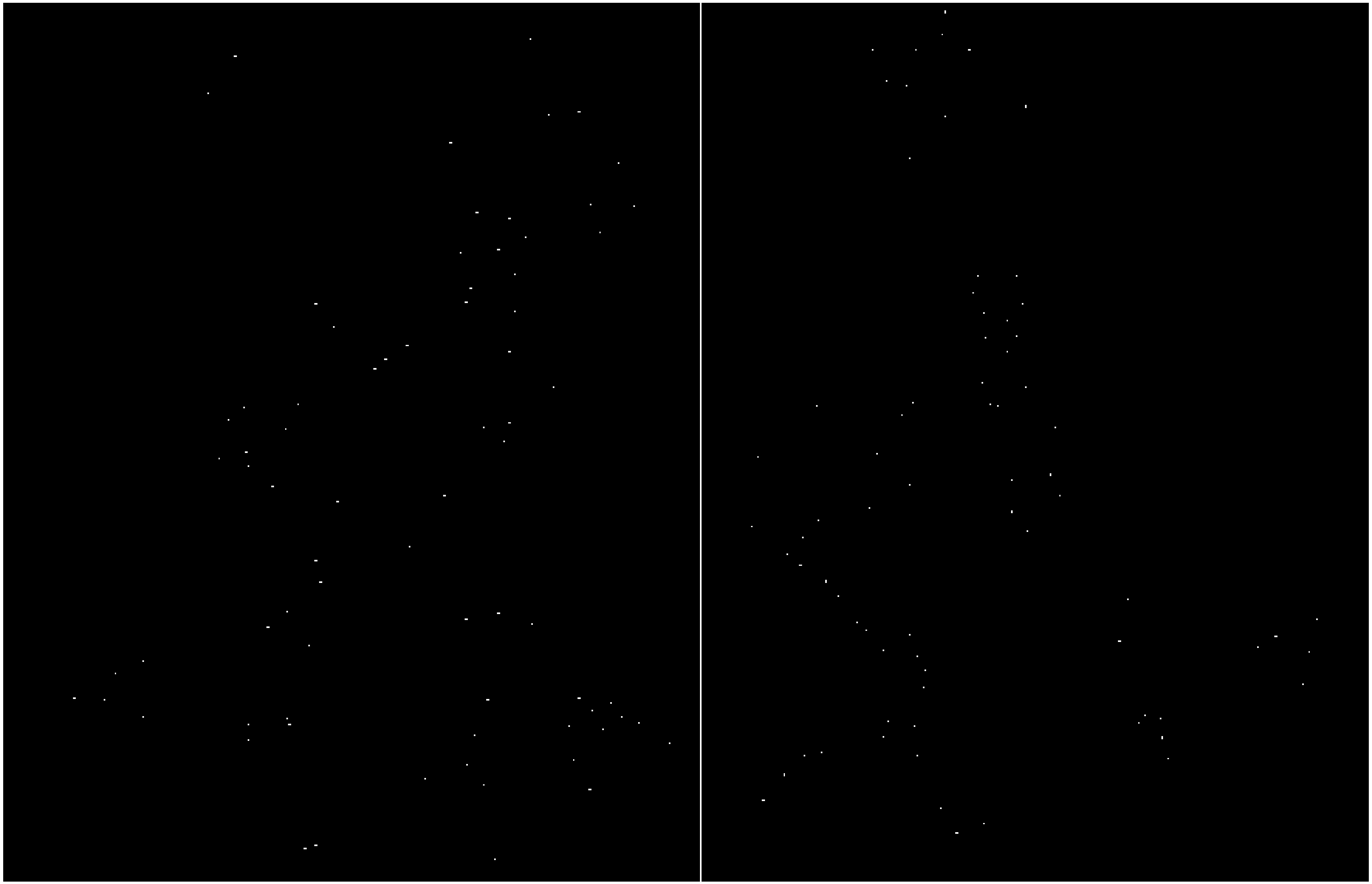
Computer cornerness scores (red high, blue low)



Threshold (Mc > value)



Find local maxima of Mc (Nonmax Suppression)



Harris features (in red)

Harris Corner Detector: Variants

- Harris and Stephens '88 is rotationally invariant and downweights edge-like features where $\lambda_1 >> \lambda_0$.

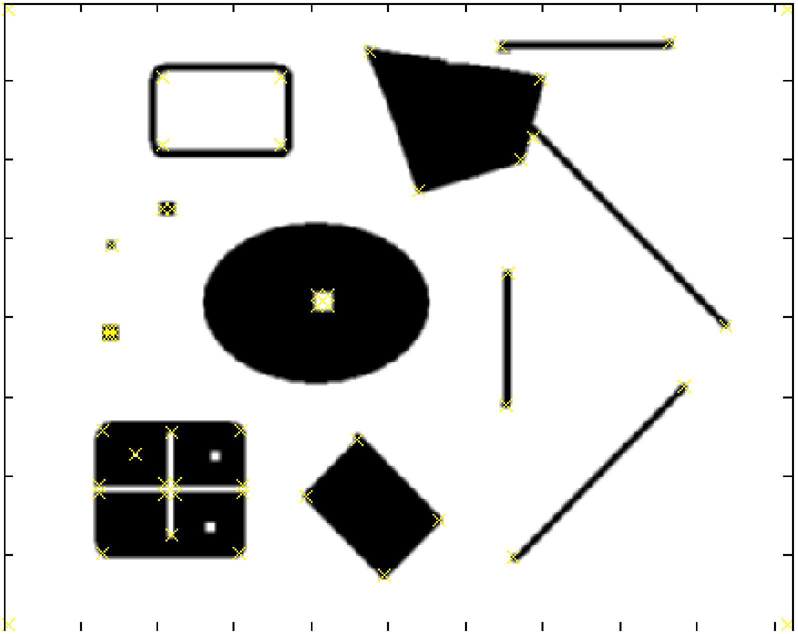
$$det(\mathbf{A}) - \alpha trace(\mathbf{A})^2 = \lambda_0\lambda_1 - \alpha(\lambda_0 + \lambda_1)^2$$

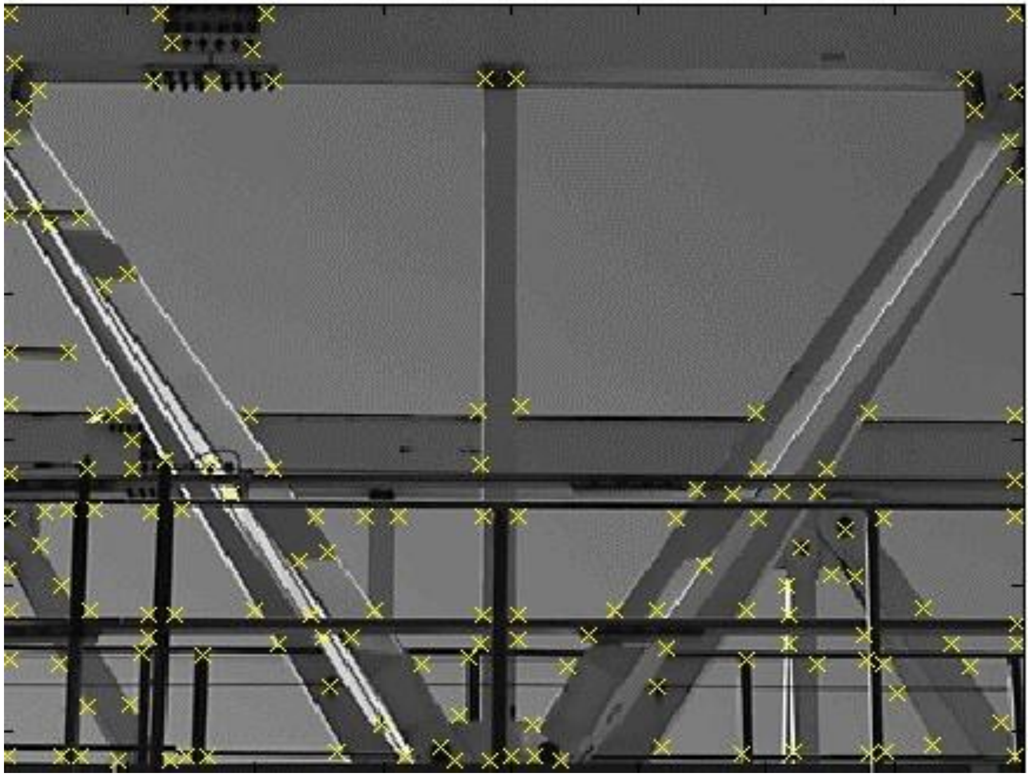
- Triggs '04 suggested $\lambda_0 - \alpha\lambda_1$.
- Brown et al, '05 use harmonic mean:

$$\frac{det(\mathbf{A})}{trace(\mathbf{A})} = \frac{\lambda_0\lambda_1}{\lambda_0 + \lambda_1}$$

which is smoother when $\lambda_0 \approx \lambda_1$

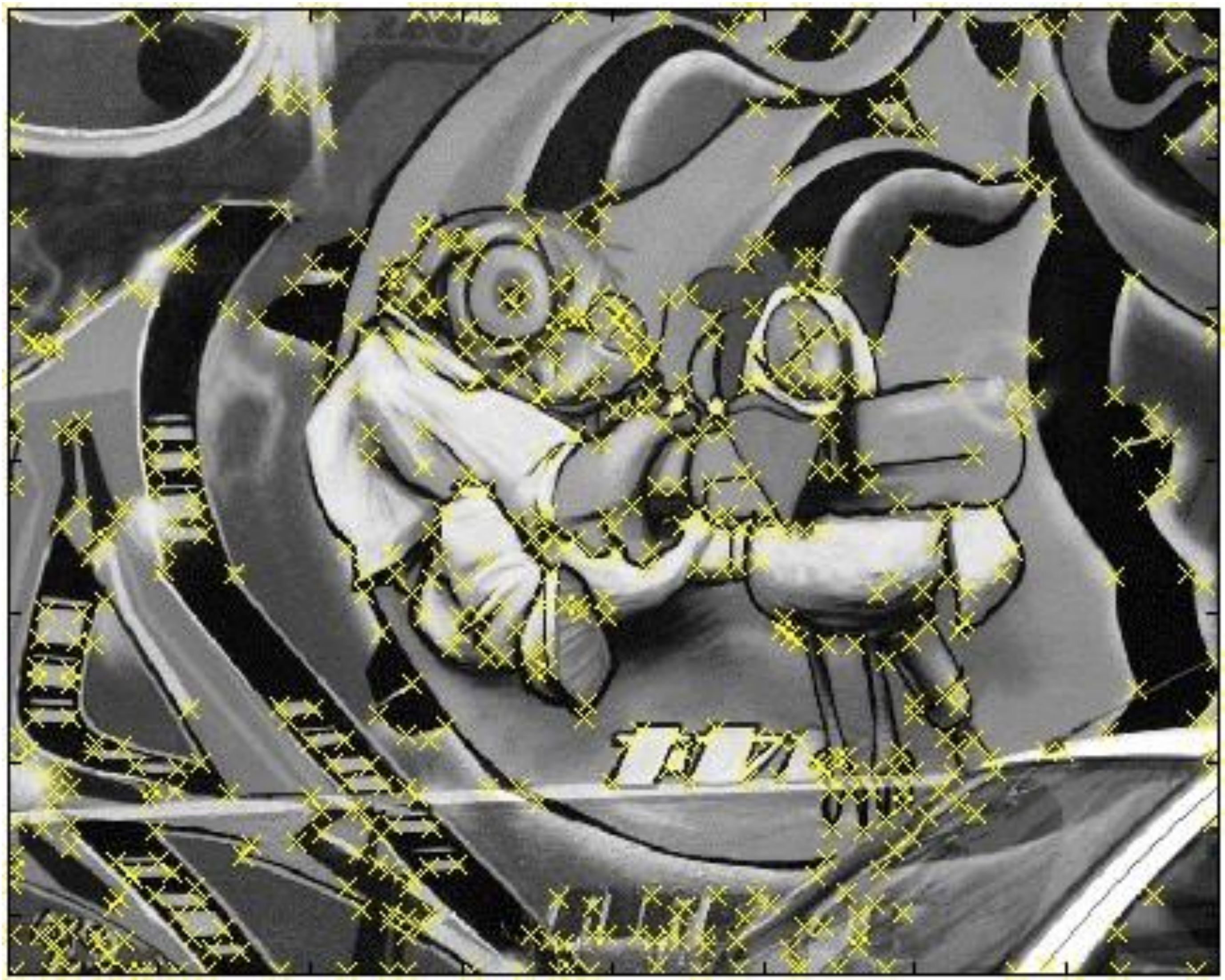
Harris Detector – Responses [Harris88]





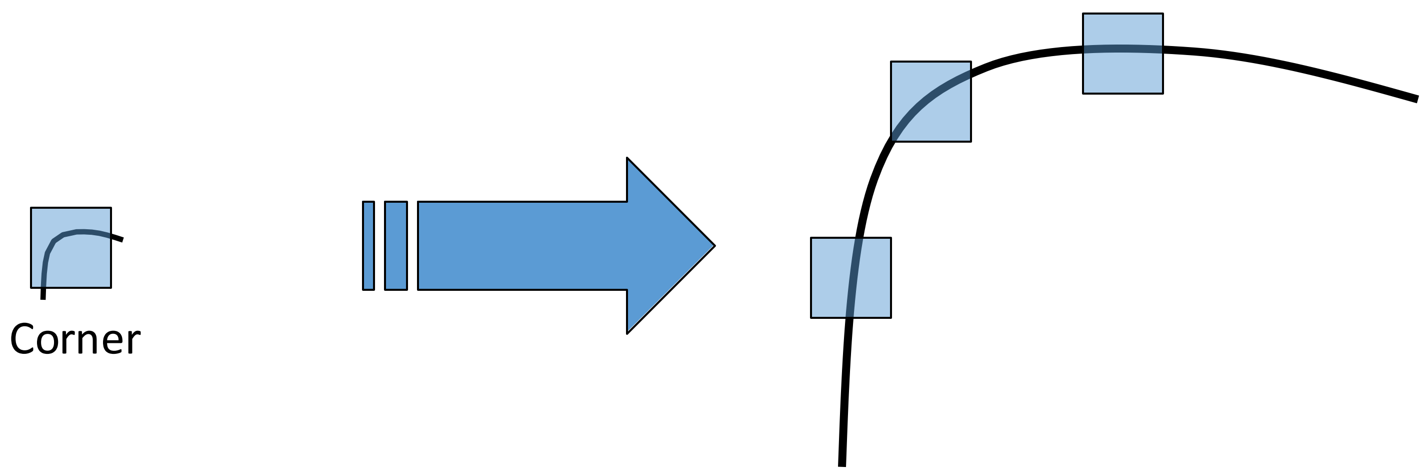
Effect: A very precise corner detector.

Harris Detector – Responses [Harris88]



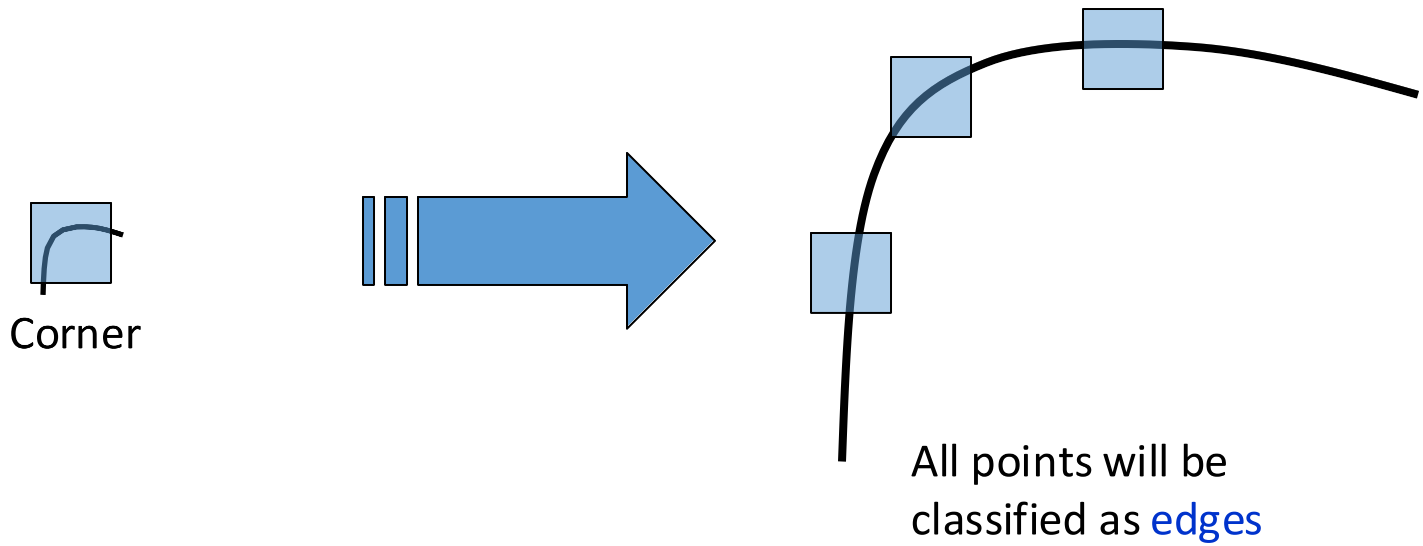
Harris Detector: Invariance Properties

- Scaling



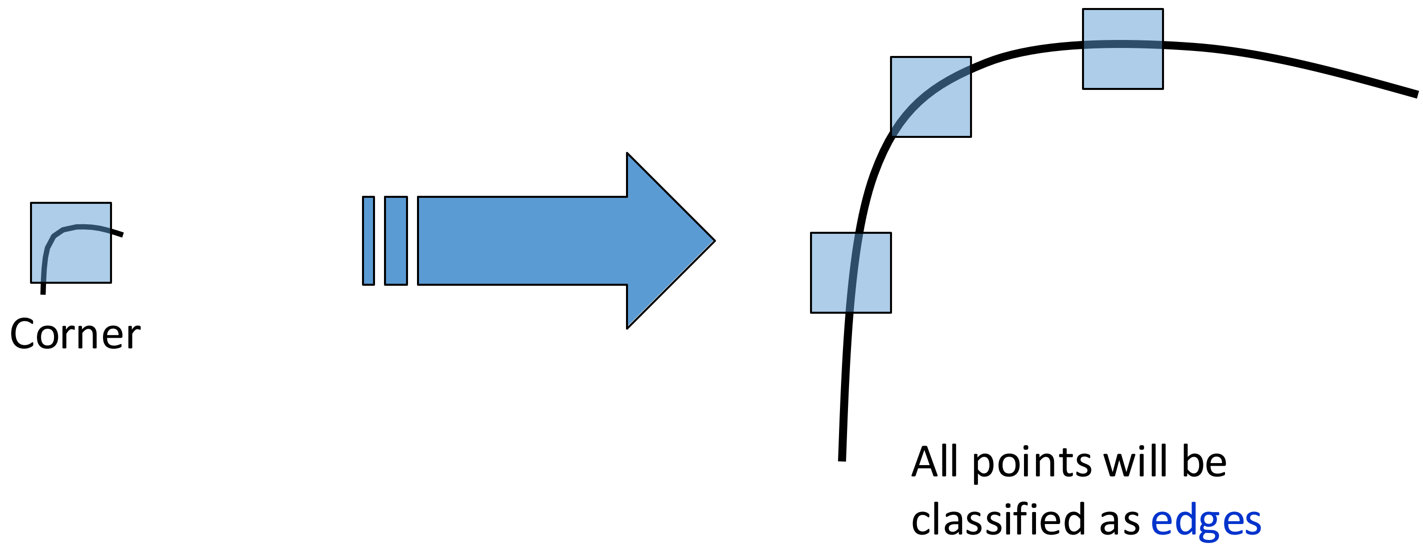
Harris Detector: Invariance Properties

- Scaling



Harris Detector: Invariance Properties

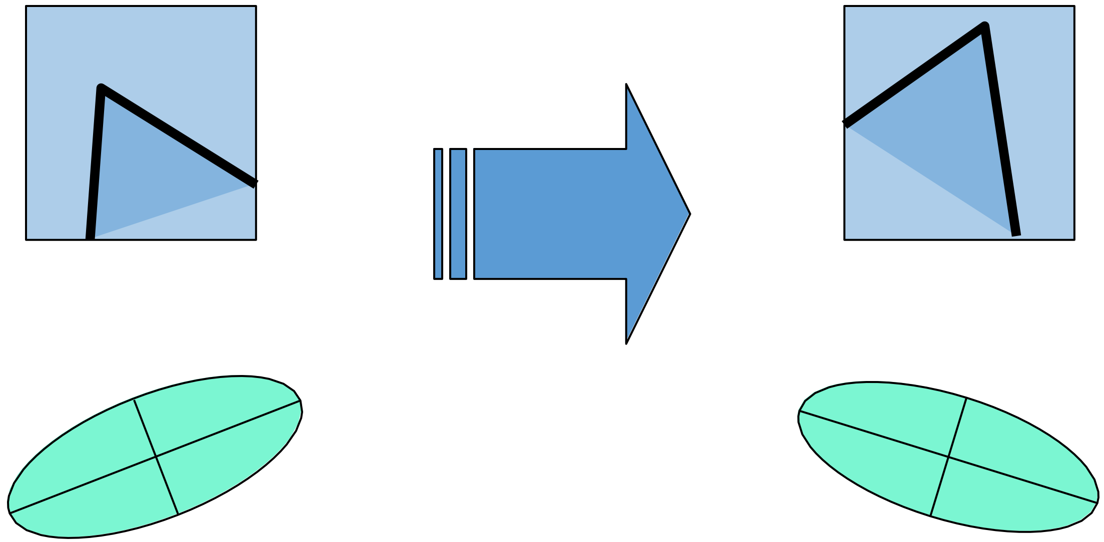
- Scaling



Not invariant to scaling

Harris Detector: Invariance Properties

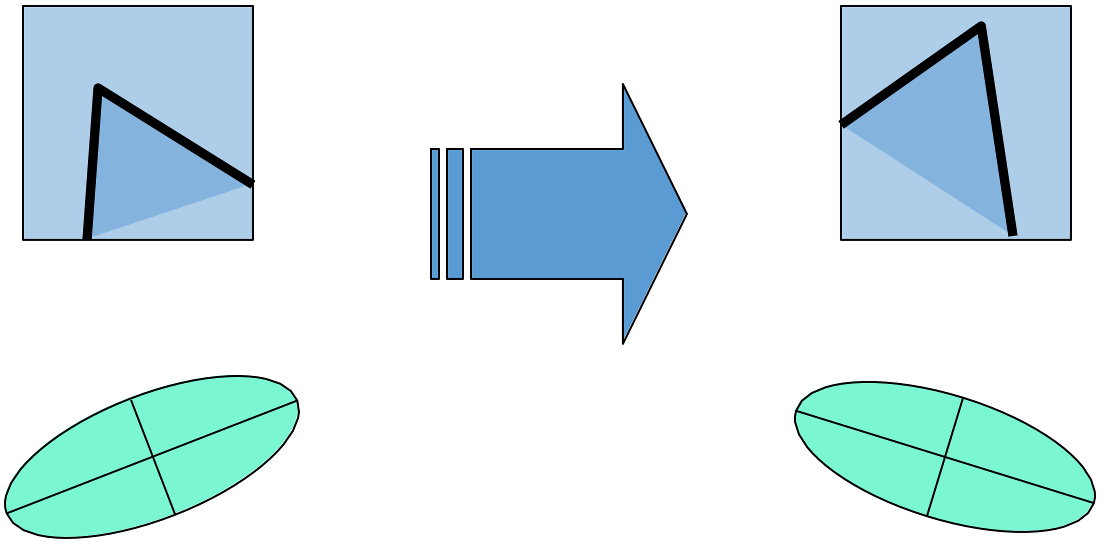
- Rotation



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

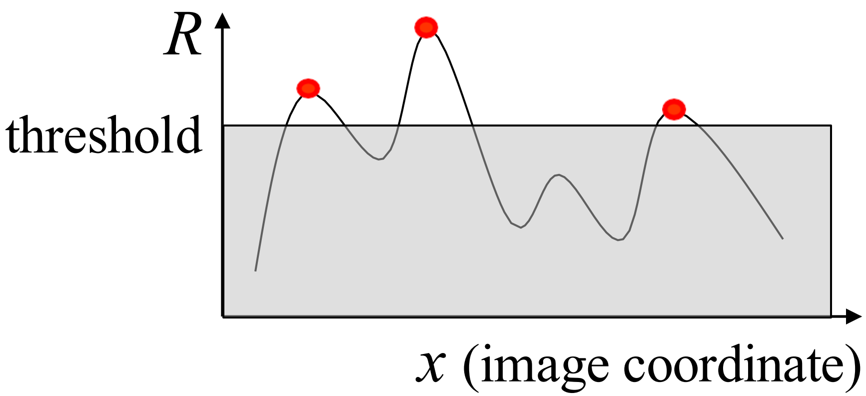
- Photometric change: Affine intensity change: $I \rightarrow aI + b$
  - ✓ Only derivatives are used =>
    invariance to intensity shift $I \rightarrow I + b$

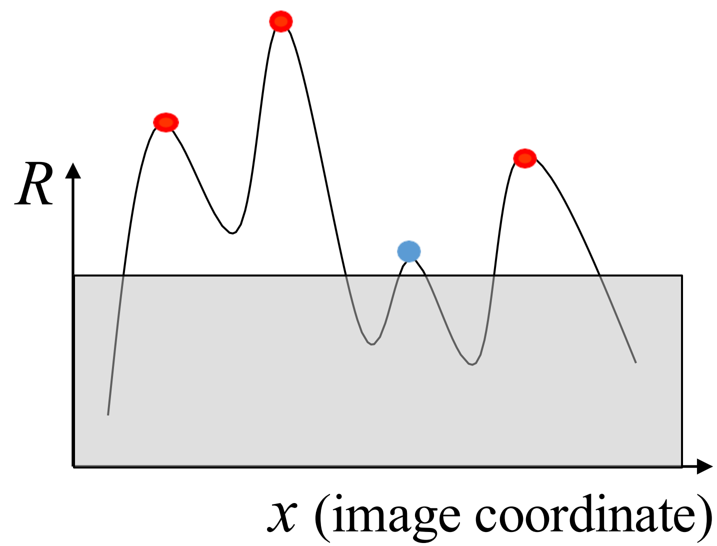
Harris Detector: Invariance Properties

- Photometric change: Affine intensity change: $I \rightarrow aI + b$

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

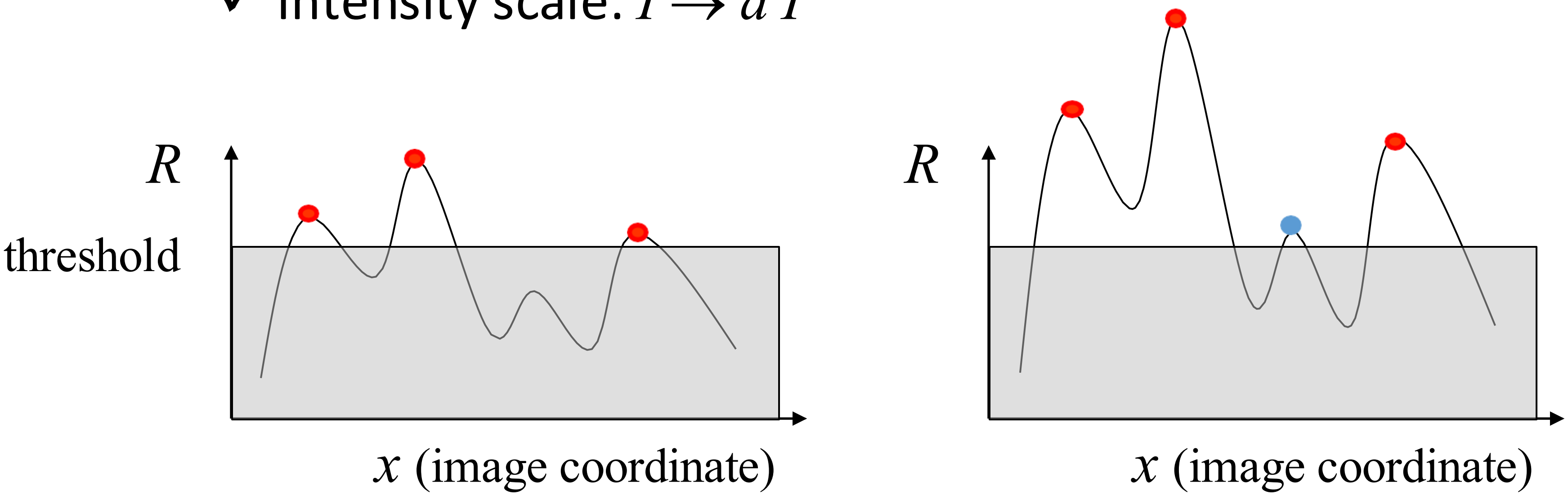
  ✓ Intensity scale: $I \rightarrow a\,I$





Harris Detector: Invariance Properties

- Photometric change: 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

Things to remember

- Keypoint detection: repeatable and distinctive
  - Corners,
  - Invariant to scale, rotation, etc.

- Harris Corner Detection
  - Rotation Invariant
  - Partial Intensity Change Invariant
  - *Not Invariant to Scale*