# Computer Vision
# Interest Points

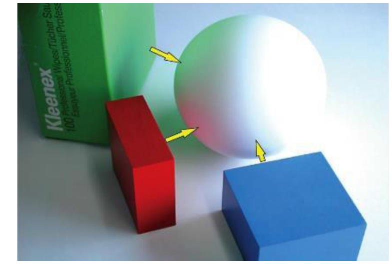
**Dr. Mrinmoy Ghorai**

**Indian Institute of Information Technology**
**Sri City, Chittoor**

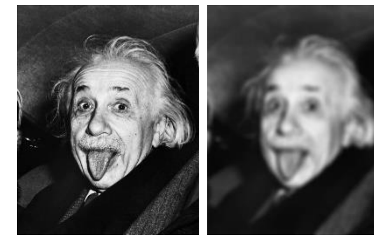
# Interest Points

# What have we learned so far?

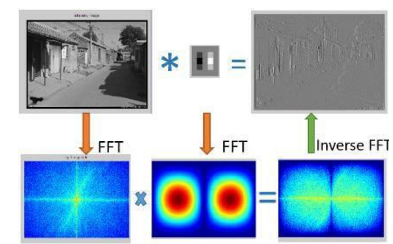
- Light and color
  – What an image records



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

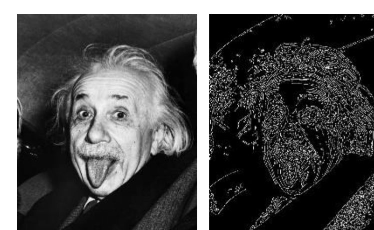


- Filtering in frequency domain
  - Filtering = change frequency of the input image
  - Denoising, sampling, image compression




- 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?

- Corner detection

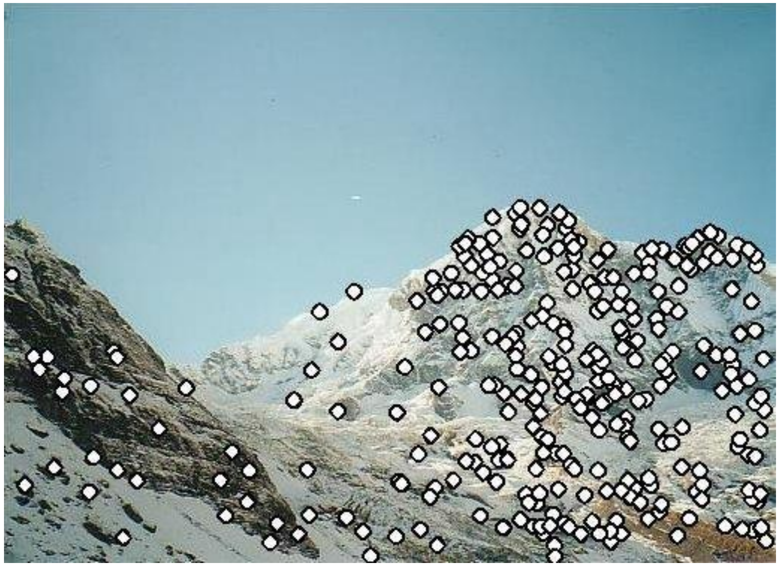
- Handling scale and orientation

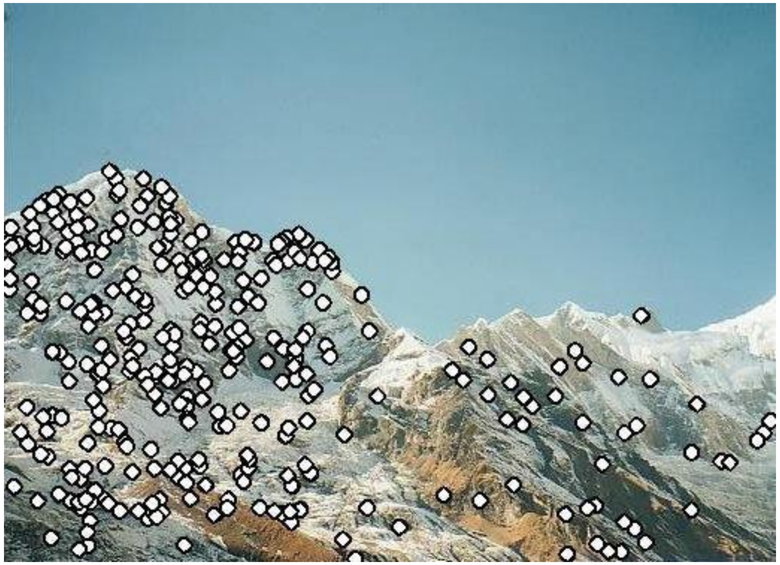
# Why extract features?

- 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?

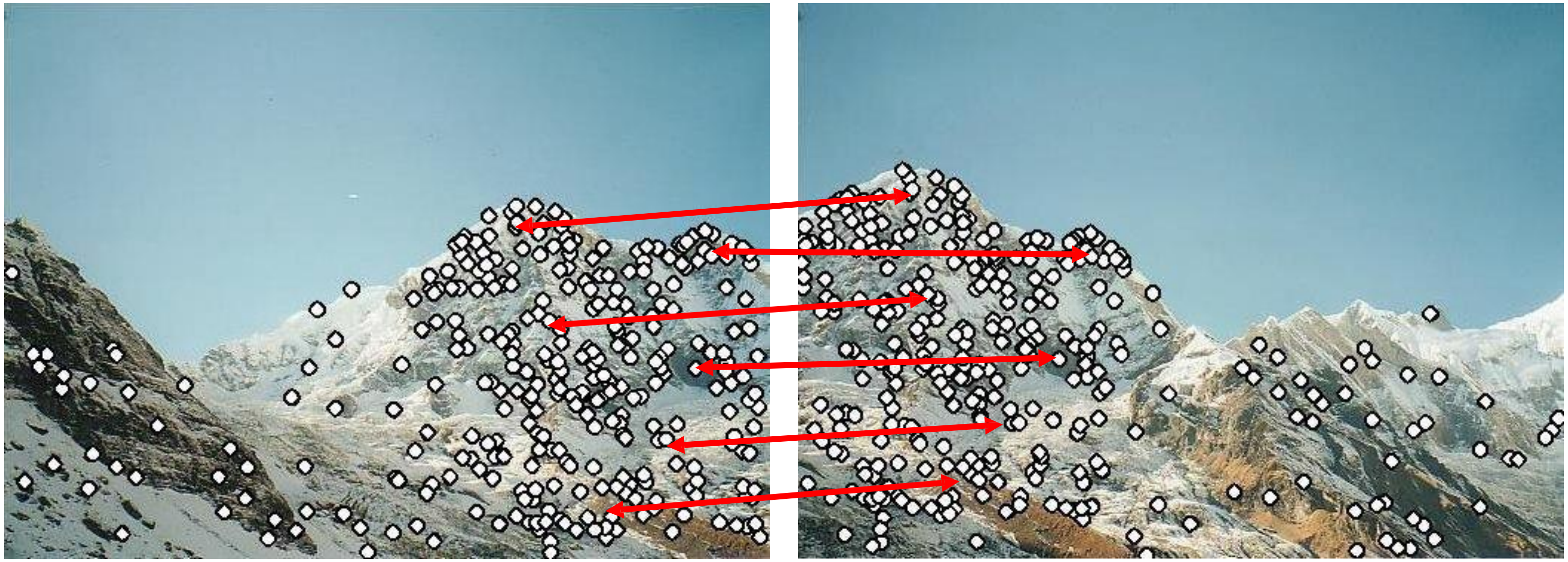




Step 1: extract features

# 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

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

# Advantages of local features

Locality

- features are local, so robust to occlusion and clutter

Quantity

- hundreds or thousands in a single image

Distinctiveness:

- can differentiate a large database of objects

Efficiency

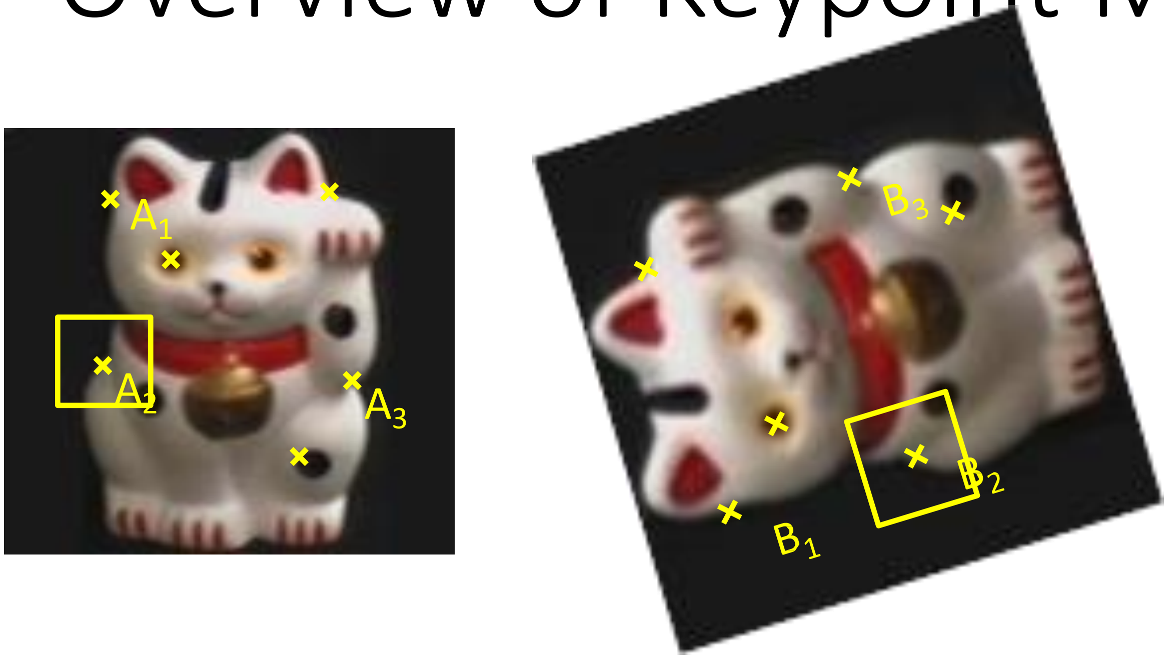
- real-time performance achievable

# Overview of Keypoint Matching



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

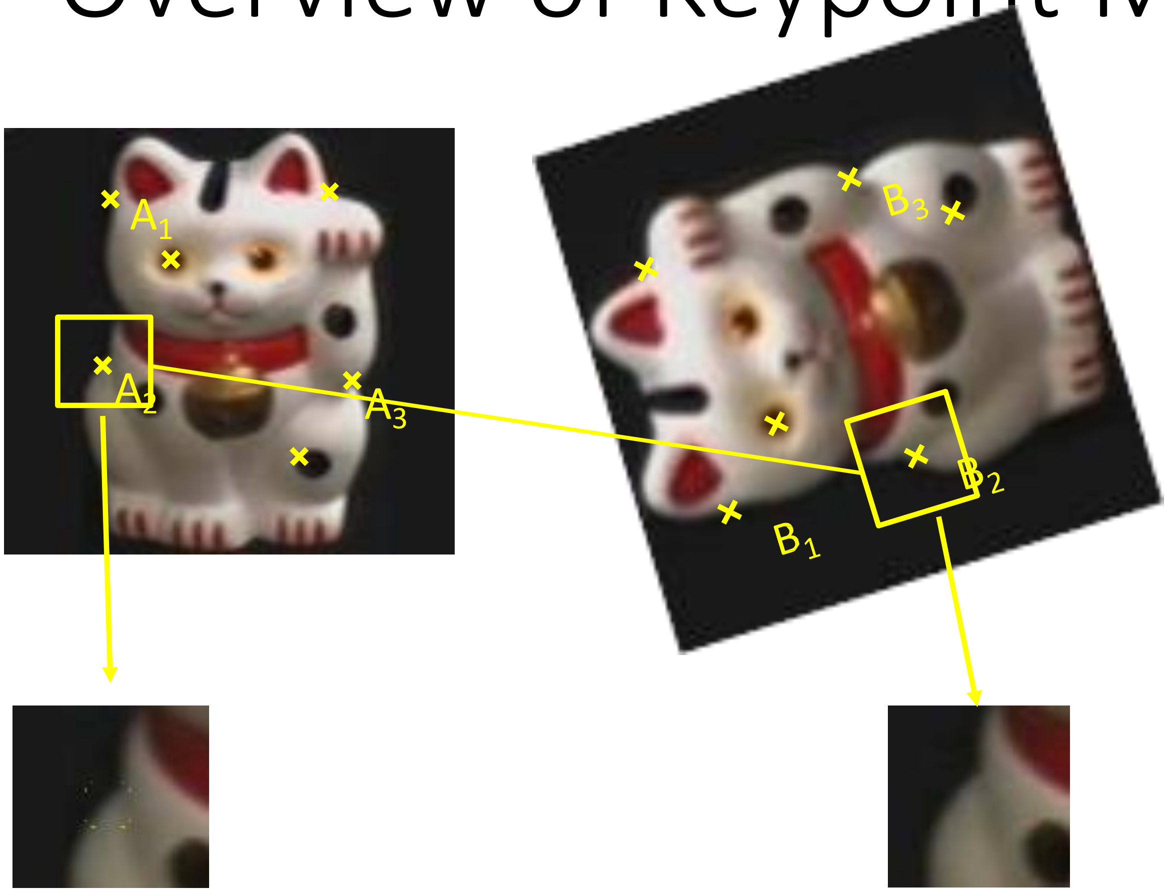
# Overview of Keypoint Matching




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

K. Grauman, B. Leibe

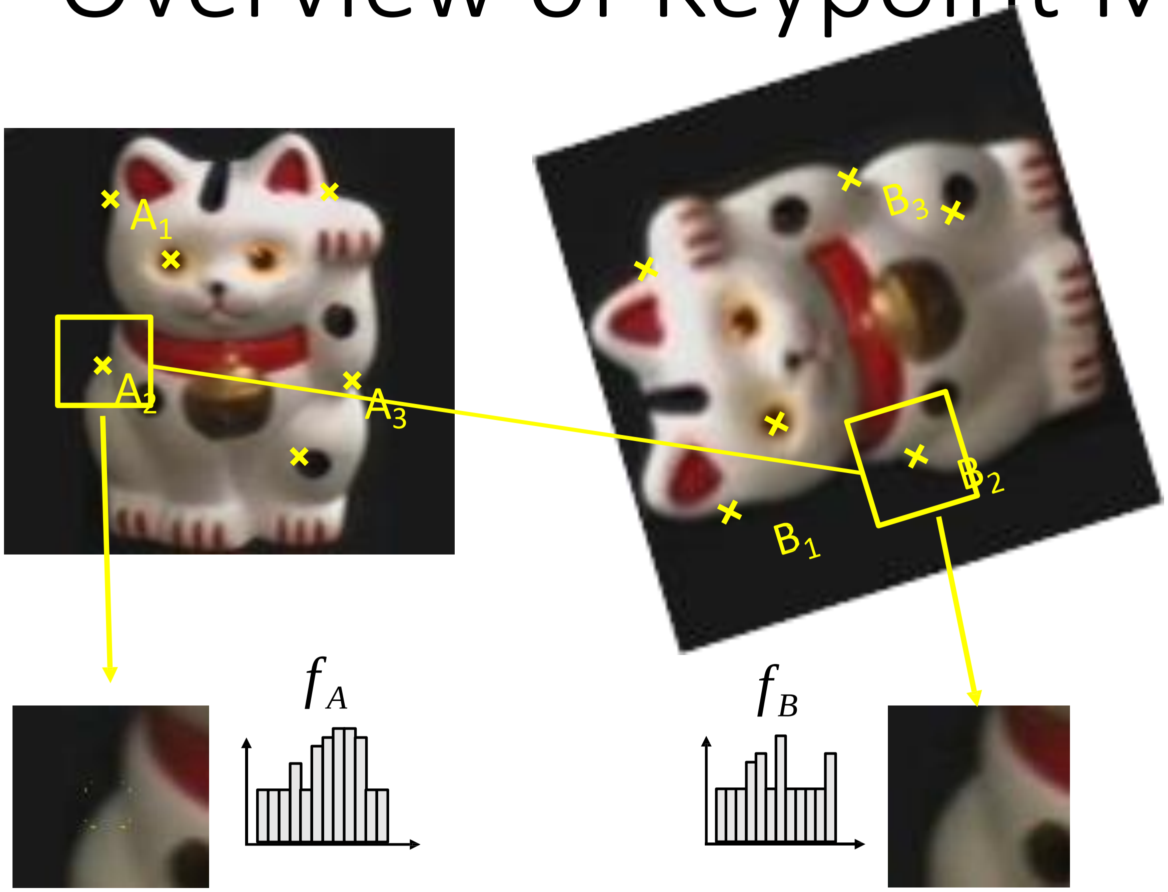
# Overview of Keypoint Matching




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

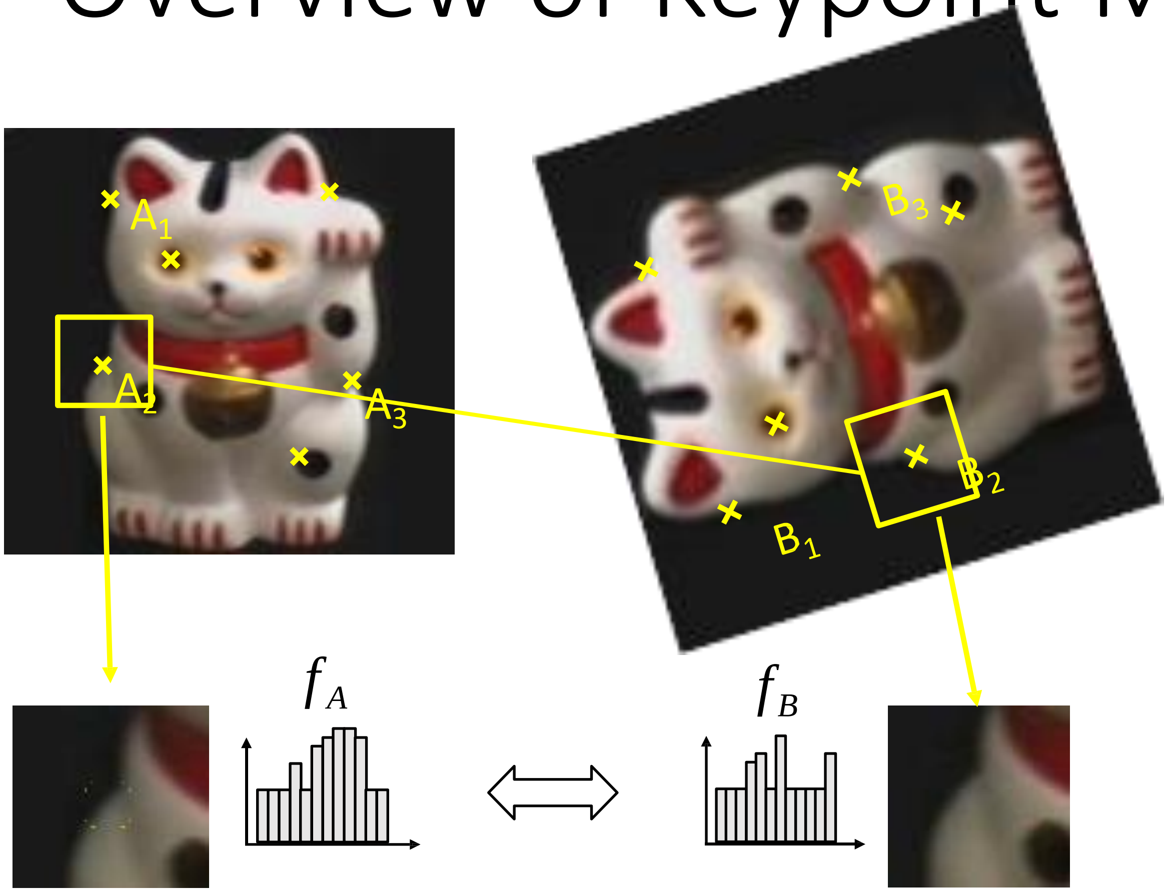
K. Grauman, B. Leibe

# 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**

# Overview of Keypoint Matching




$$d(f_A, f_B) < T$$

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**

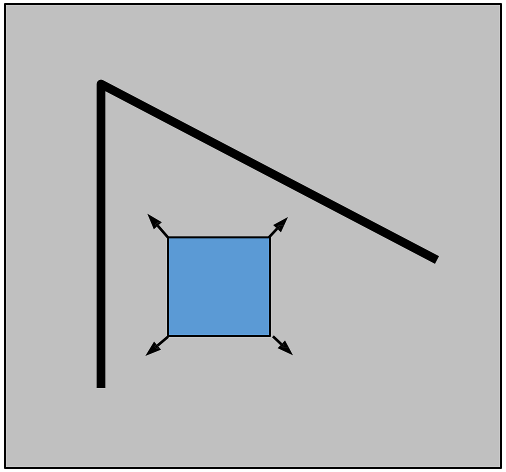
K. Grauman, B. Leibe

# Goals for Keypoints

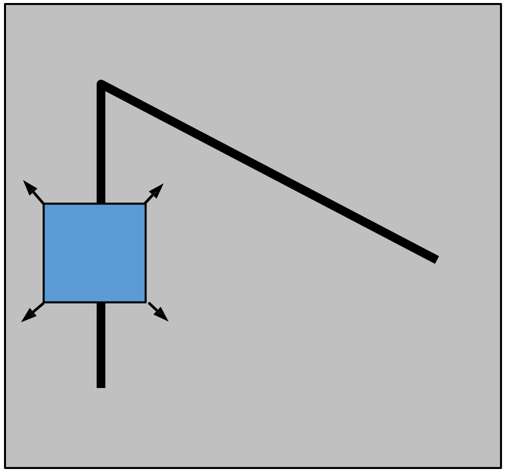
Detect points that are *repeatable* and *distinctive*

# Corner Detection: Basic Idea

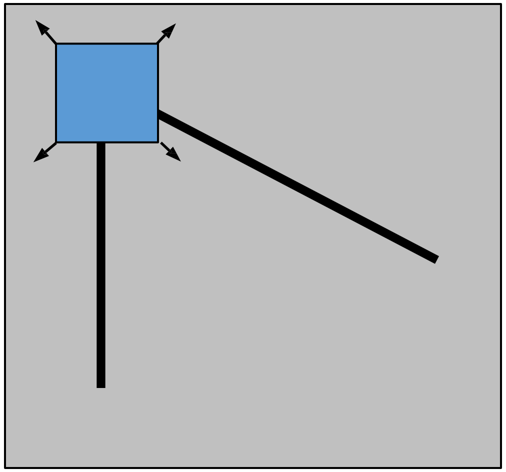
- How does the window change when you shift it?
- Shifting the window in any direction causes a big change



"flat" region:
no change in
all directions



"edge":
no change along the
edge direction

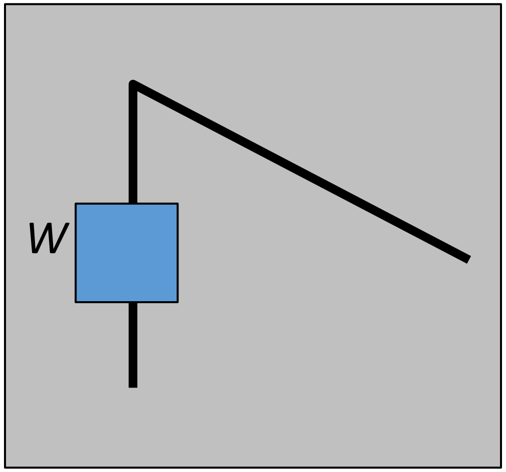


"corner":
significant change in
all directions

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

# Corner detection: the math

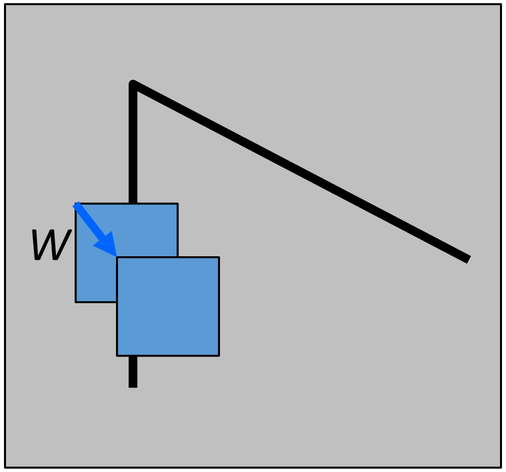
Consider shifting the window $W$ by $(u,v)$



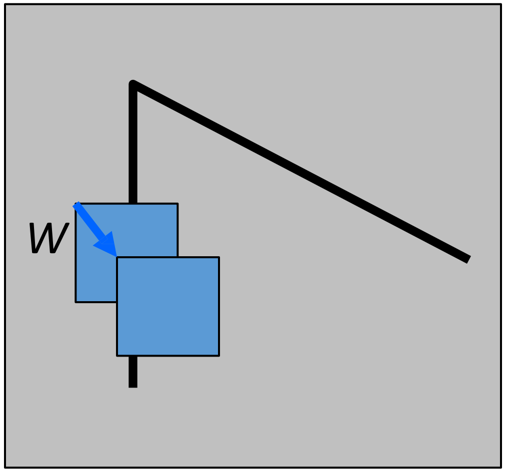
# Corner detection: the math

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

- how do the pixels in W change?



# Corner detection: the math



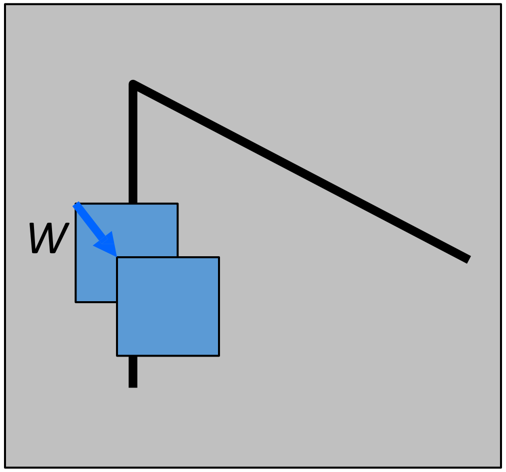

Consider shifting the window *W* by (*u*,*v*)

- 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*):




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

# 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*:

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

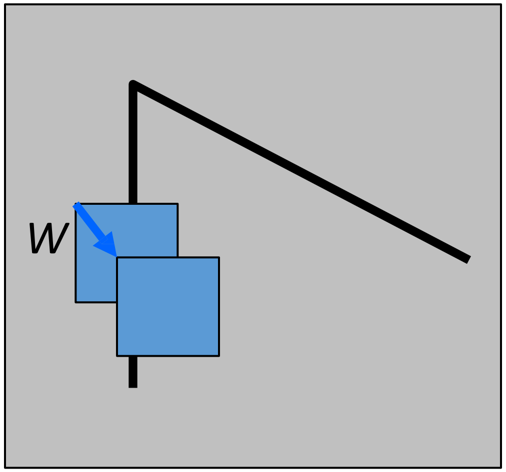
If the motion (u,v) is small, then first order approximation is good

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

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

Plugging this into the formula on the previous slide…

# Corner detection: the math

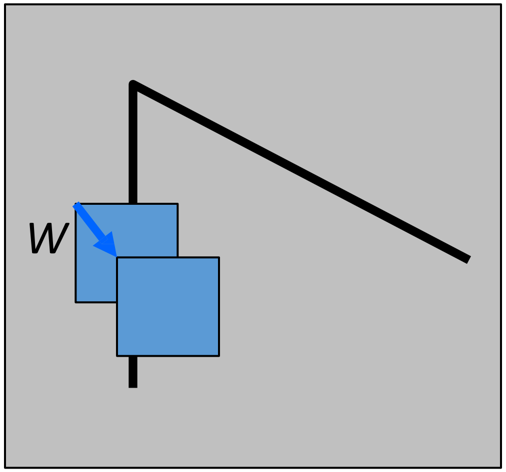



Using the small motion assumption,
replace $I$ with a linear approximation

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

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

# Corner detection: the math



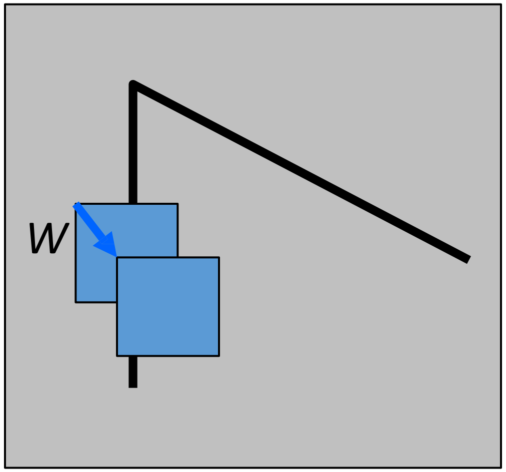

Using the small motion assumption,
replace $I$ with a linear approximation

(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

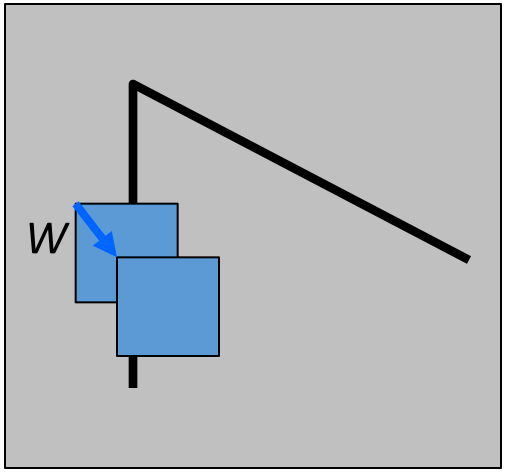
(Shorthand: $I_x = \frac{\partial I}{\partial x}$ )

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

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

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

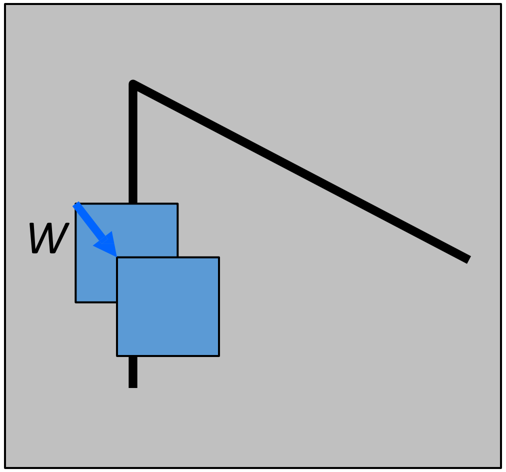
# Corner detection: the math




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

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

# Corner detection: the math



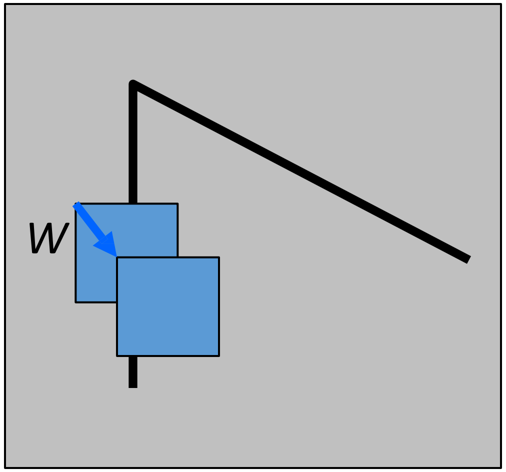

$$E(u,v) \approx \sum_{(x,y)\in W} (I_x(x,y)u + I_y(x,y)v)^2$$

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

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

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

# 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.

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

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

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

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

# The second moment matrix

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

$$
\begin{aligned}
E(u,v) &\approx Au^2 + 2Buv + Cv^2 \\
&\approx \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}
\end{aligned}
$$

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

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

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

# 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.

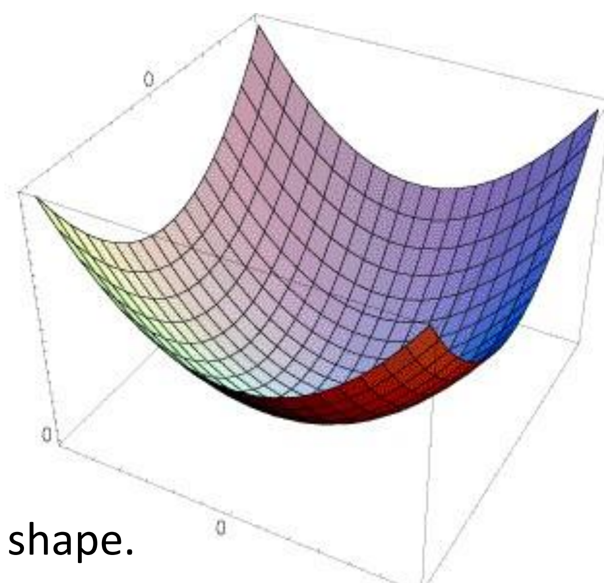
$$E(u,v) \approx Au^2 + 2Buv + Cv^2$$

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

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

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

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



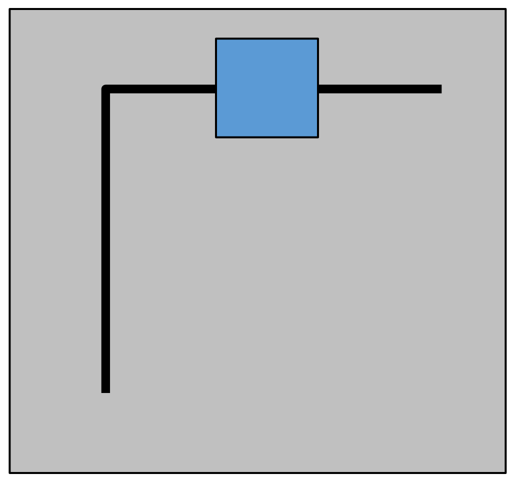

Let's try to understand its shape.

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

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

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

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



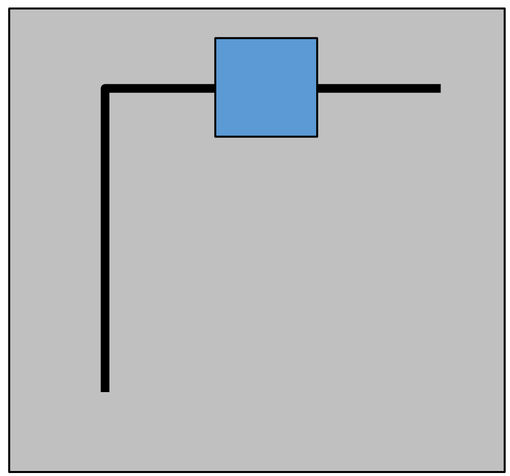
Horizontal edge: $I_x = 0$

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

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

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

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



Horizontal edge: $I_x = 0$

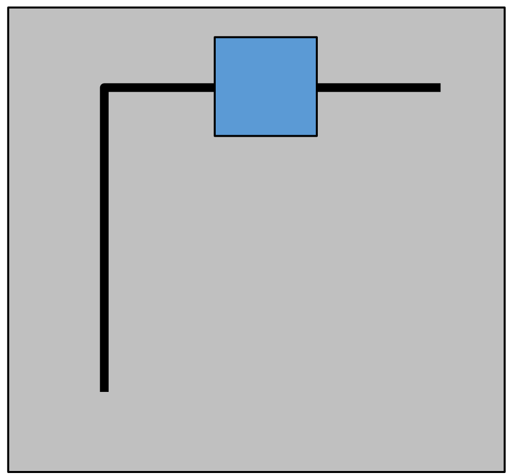
$$H = \begin{bmatrix} 0 & 0 \\ 0 & C \end{bmatrix}$$

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

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

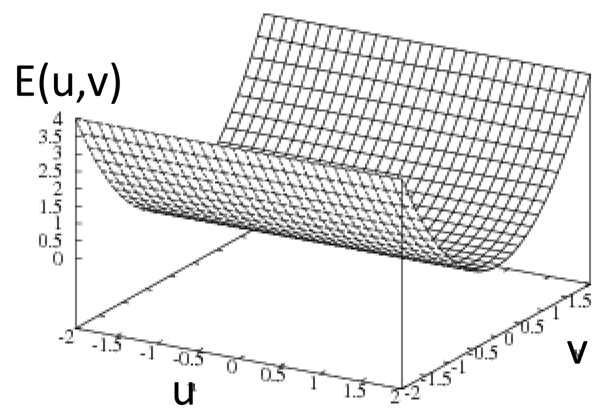
$$B = \sum_{(x,y)\in W} I_x I_y$$

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



Horizontal edge: $I_x = 0$

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

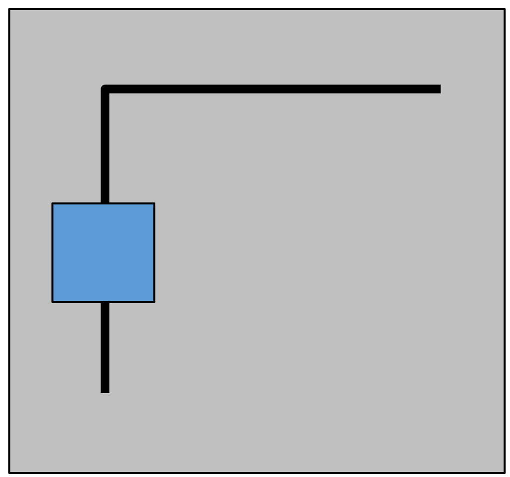


$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

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

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

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



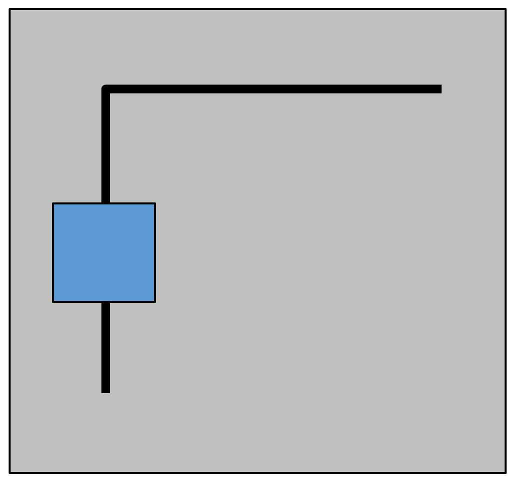
Vertical edge: $I_y = 0$

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

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

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

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



Vertical edge: $I_y = 0$

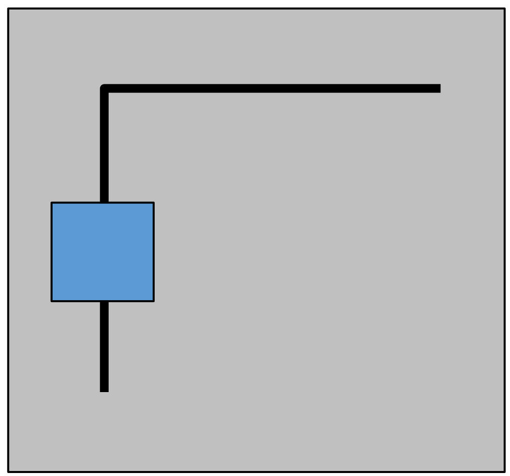
$$H = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}$$

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

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

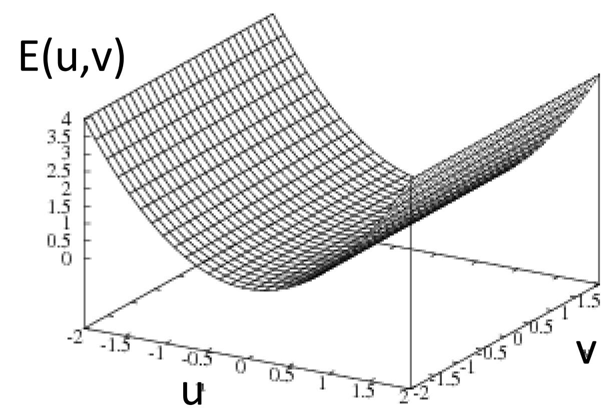
$$B = \sum_{(x,y)\in W} I_x I_y$$

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



Vertical edge: $I_y = 0$

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



# General case

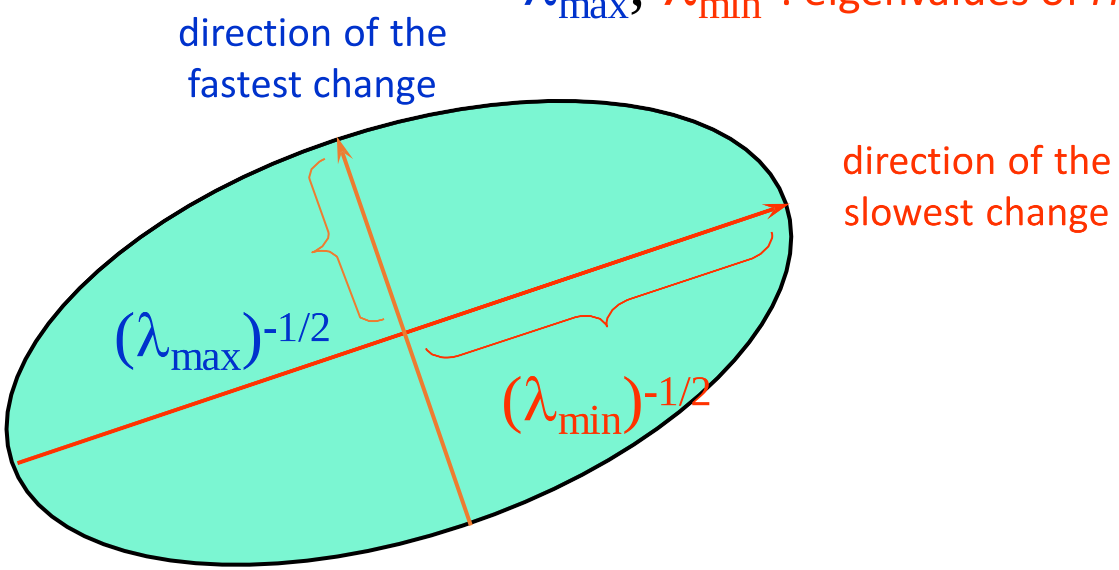
The shape of *H* tells us something about the *distribution of gradients* around a pixel

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:

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

$\lambda_{max}$, $\lambda_{min}$ : eigenvalues of *H*



# Quick eigenvalue/eigenvector review

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

The scalar $\lambda$ is the **eigenvalue** corresponding to **x**

# Quick eigenvalue/eigenvector review

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

The scalar $\lambda$ 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:

The scalar $\lambda$ 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:

The scalar $\lambda$ 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:

The scalar $\lambda$ 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:

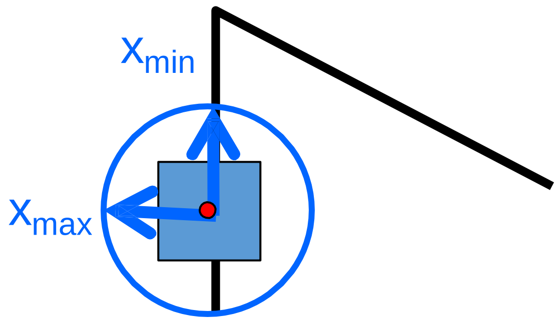
$$\lambda_{\pm} = \frac{1}{2} \left[ (h_{11} + h_{22}) \pm \sqrt{4 h_{12} h_{21} + (h_{11} - h_{22})^2} \right]$$

Once you know $\lambda$, you find **x** by solving

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

# Corner detection: the math

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




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

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

Eigenvalues and eigenvectors of H

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

# Corner detection: the math

How are $\lambda_{max}$, $x_{max}$, $\lambda_{min}$, and $x_{min}$ relevant for feature detection?

- What's our feature scoring function?

# Corner detection: the math

How are $\lambda_{max}$, $x_{max}$, $\lambda_{min}$, 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 ($\lambda_{min}$) of $H$

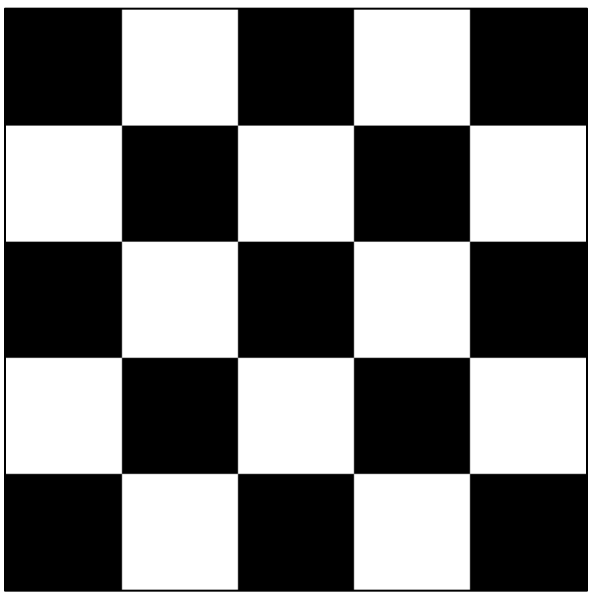
# Corner detection: the math

How are $\lambda_{max}$, $x_{max}$, $\lambda_{min}$, 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 ($\lambda_{min}$) of $H$



$I$

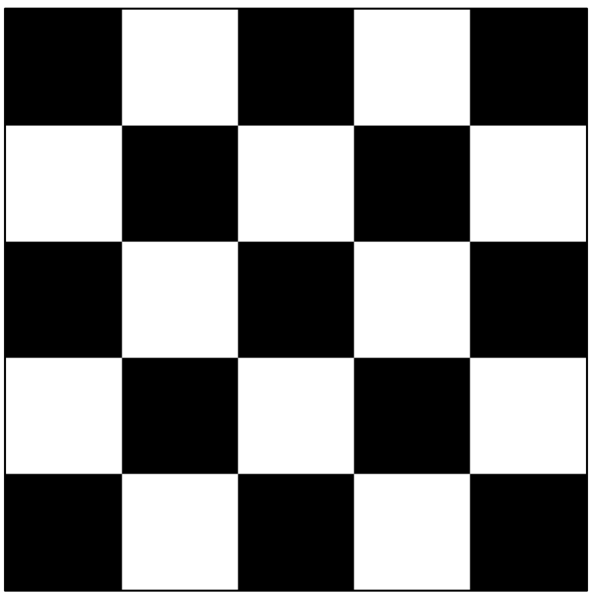
# Corner detection: the math

How are $\lambda_{max}$, $x_{max}$, $\lambda_{min}$, and $x_{min}$ relevant for feature detection?

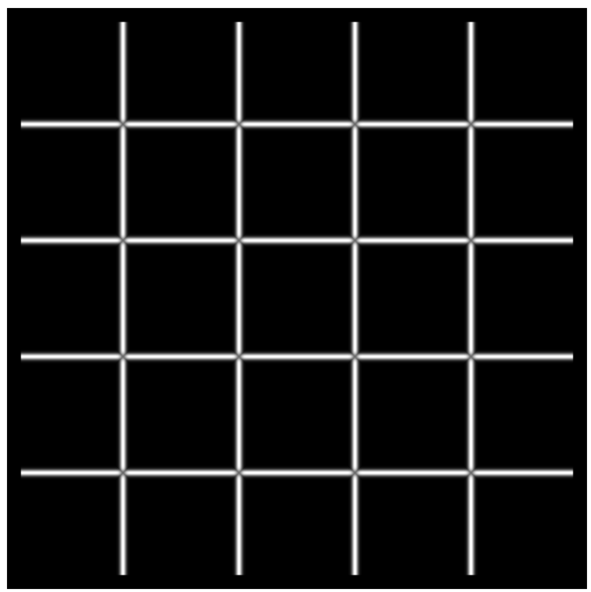
- What's our feature scoring function?

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 ($\lambda_{min}$) of $H$



$I$



$\lambda_{\max}$

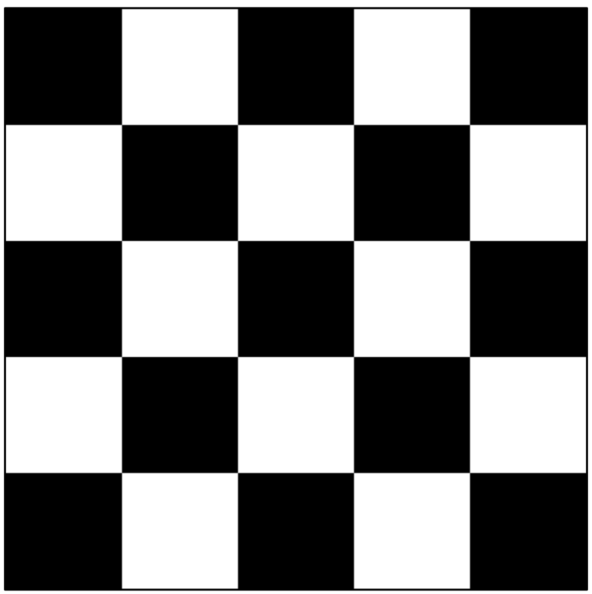
# Corner detection: the math

How are $\lambda_{max}$, $x_{max}$, $\lambda_{min}$, and $x_{min}$ relevant for feature detection?

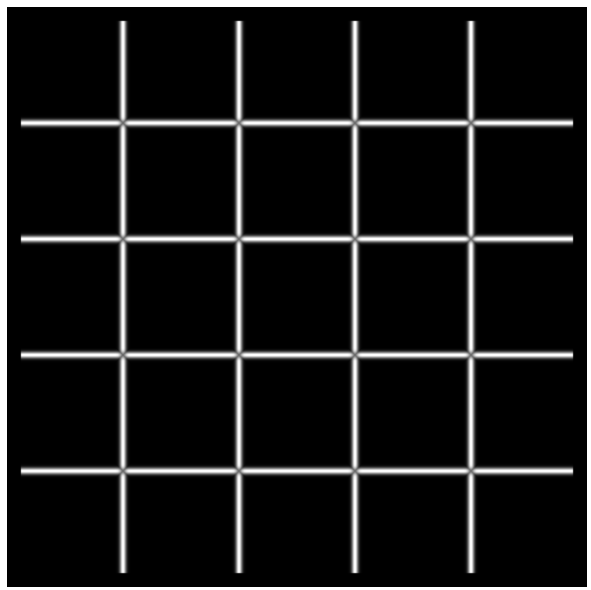
- What's our feature scoring function?

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

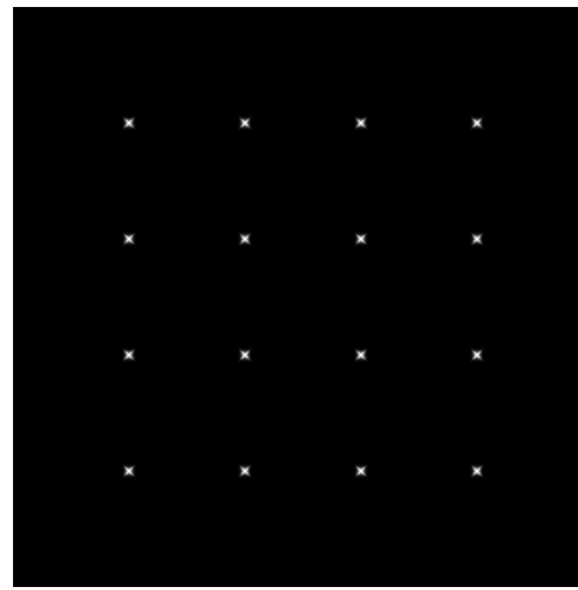
- the minimum of $E(u,v)$ should be large, over all unit vectors $[u\ v]$
- this minimum is given by the smaller eigenvalue ($\lambda_{min}$) of $H$



$I$



$\lambda_{\max}$

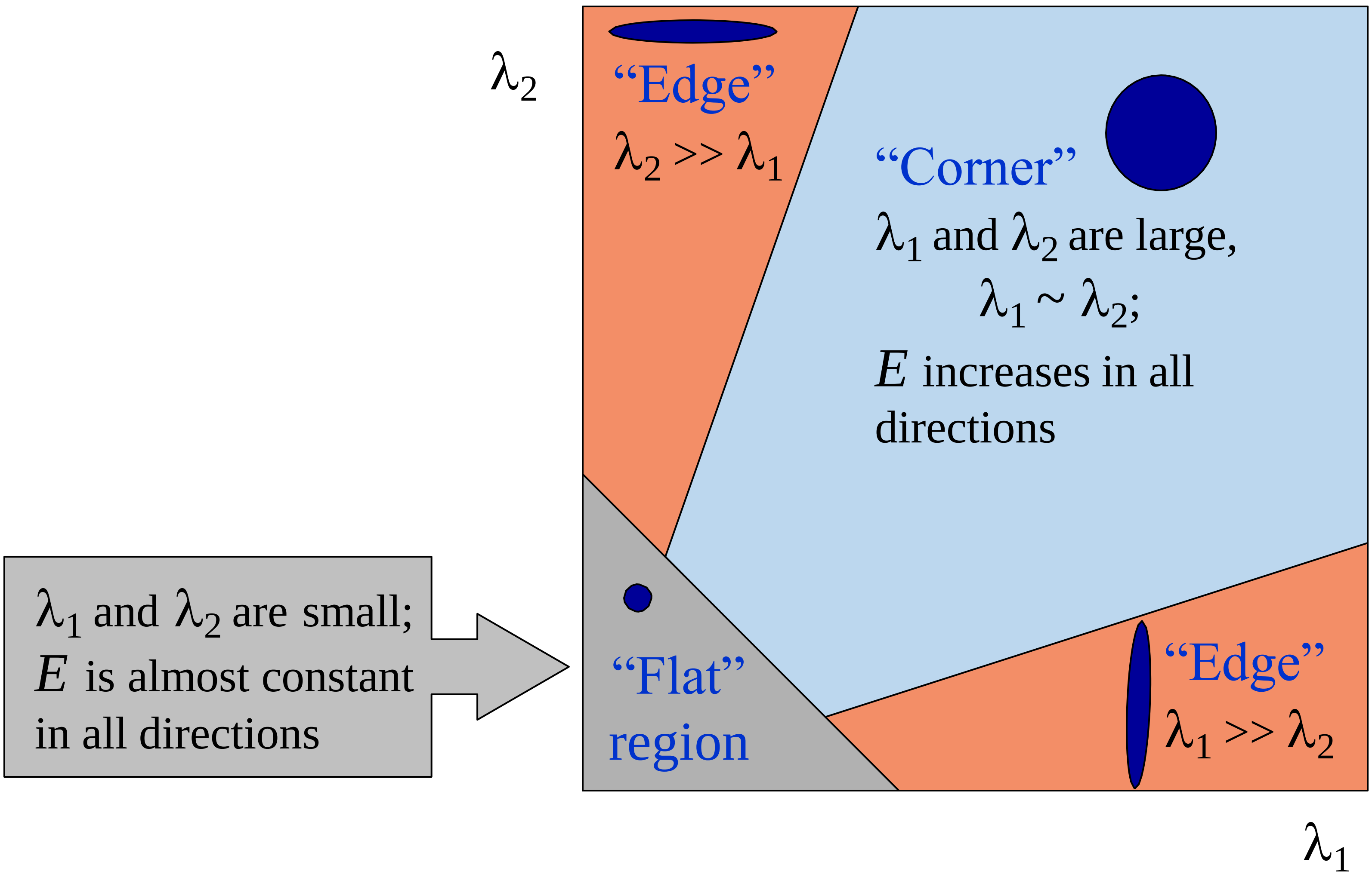


$\lambda_{\min}$

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

# Interpreting the eigenvalues

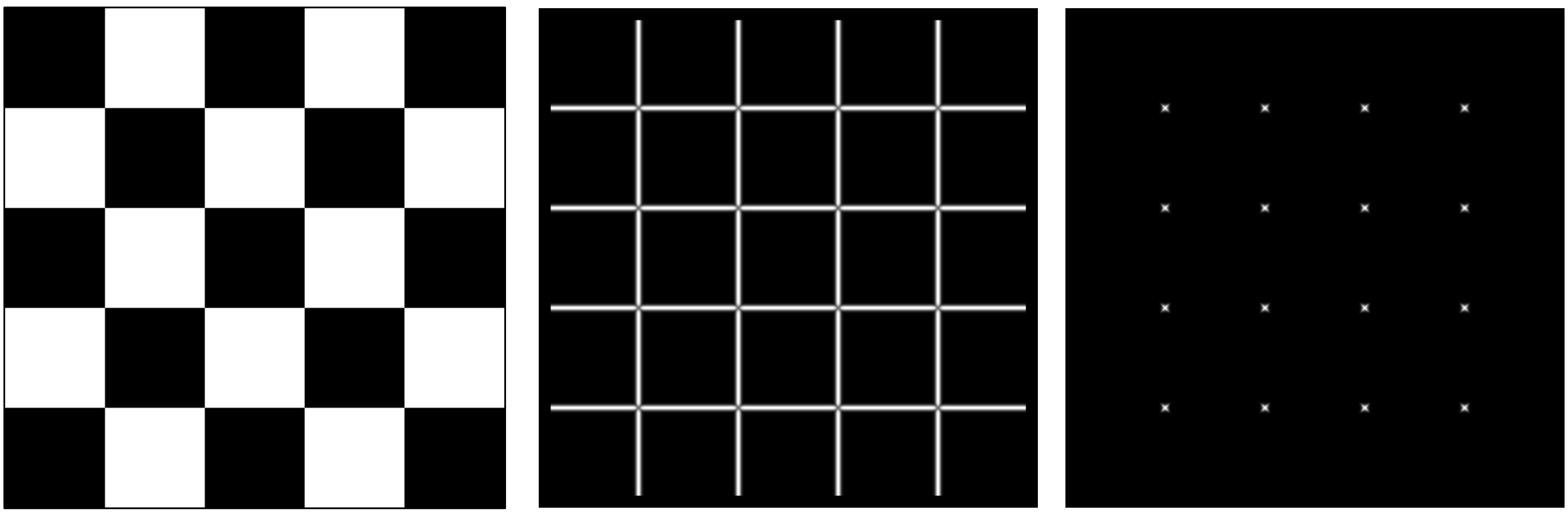
Classification of image points using eigenvalues of $M$:



# 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 ($\lambda_{min}$ > threshold)
- Choose those points where $\lambda_{min}$ is a local maximum as features



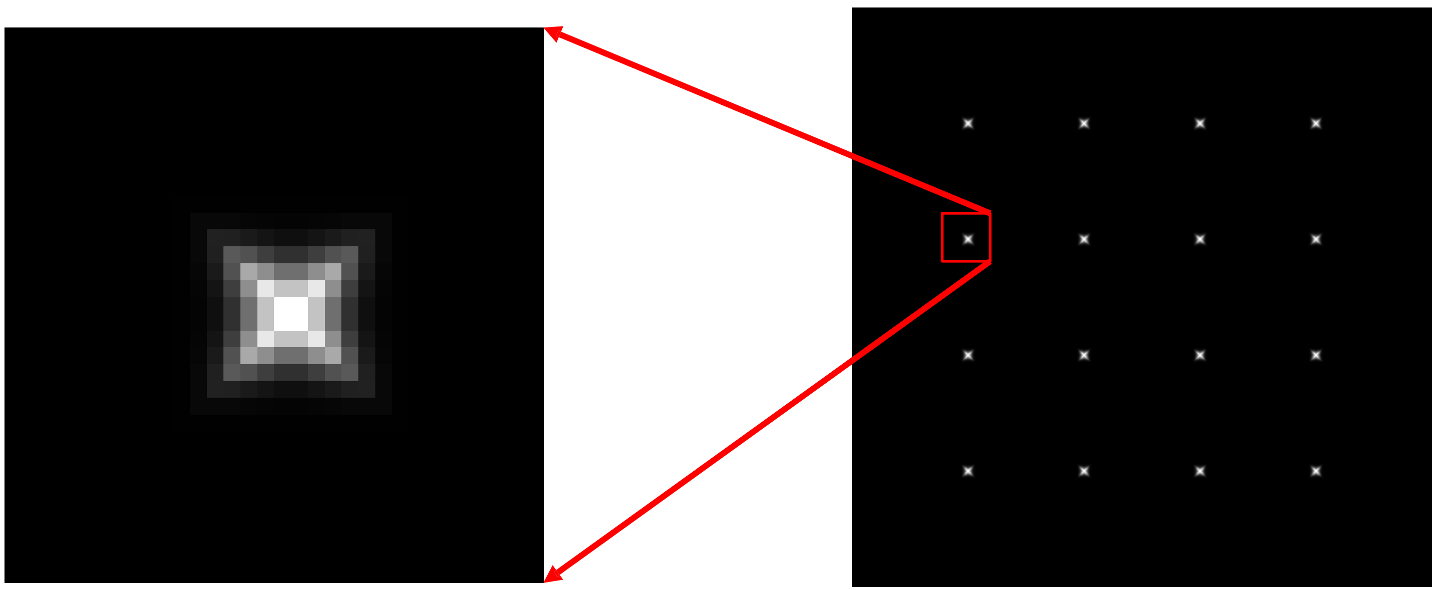
$I$ $\lambda_{\max}$ $\lambda_{\min}$

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 ($\lambda_{min}$ > threshold)
- Choose those points where $\lambda_{min}$ is a local maximum as features



$\lambda_{\min}$

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

# The Harris operator

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

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

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

- 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 combi ned cor ner and edge detector ". *Proceedings of the 4th Alvey Vision Conference*. pp. 147–151.

# Noble's corner operator

The "Noble's operator"[1] for feature detection is:

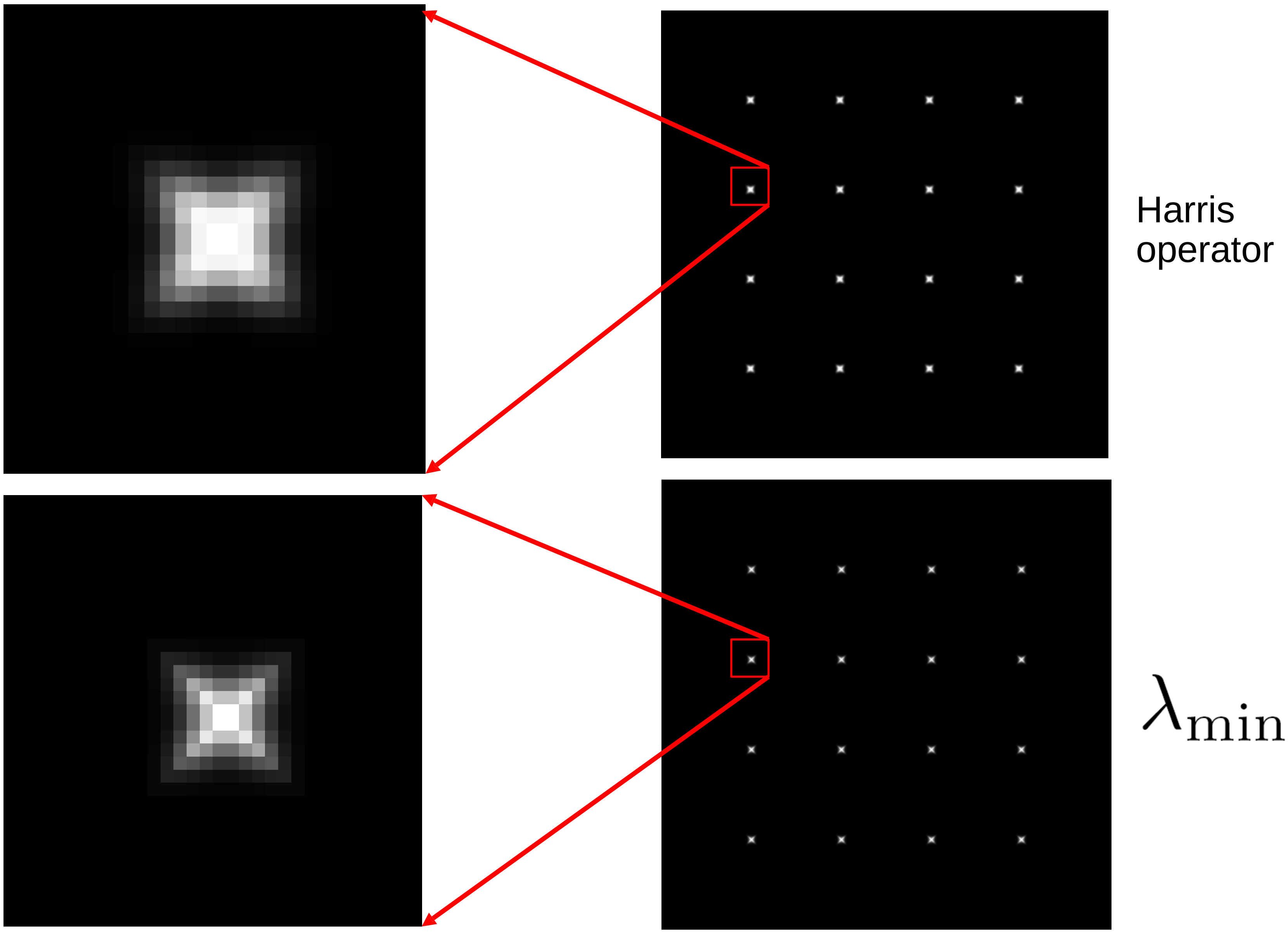
$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$

$$= \frac{determinant(H)}{trace(H)}$$

- The *trace* is the sum of the diagonals, i.e., $trace(H) = h_{11} + h_{22}$
- Very similar to $\lambda_{min}$ but less expensive (no square root)
- Called the "Noble's Corner Detector"

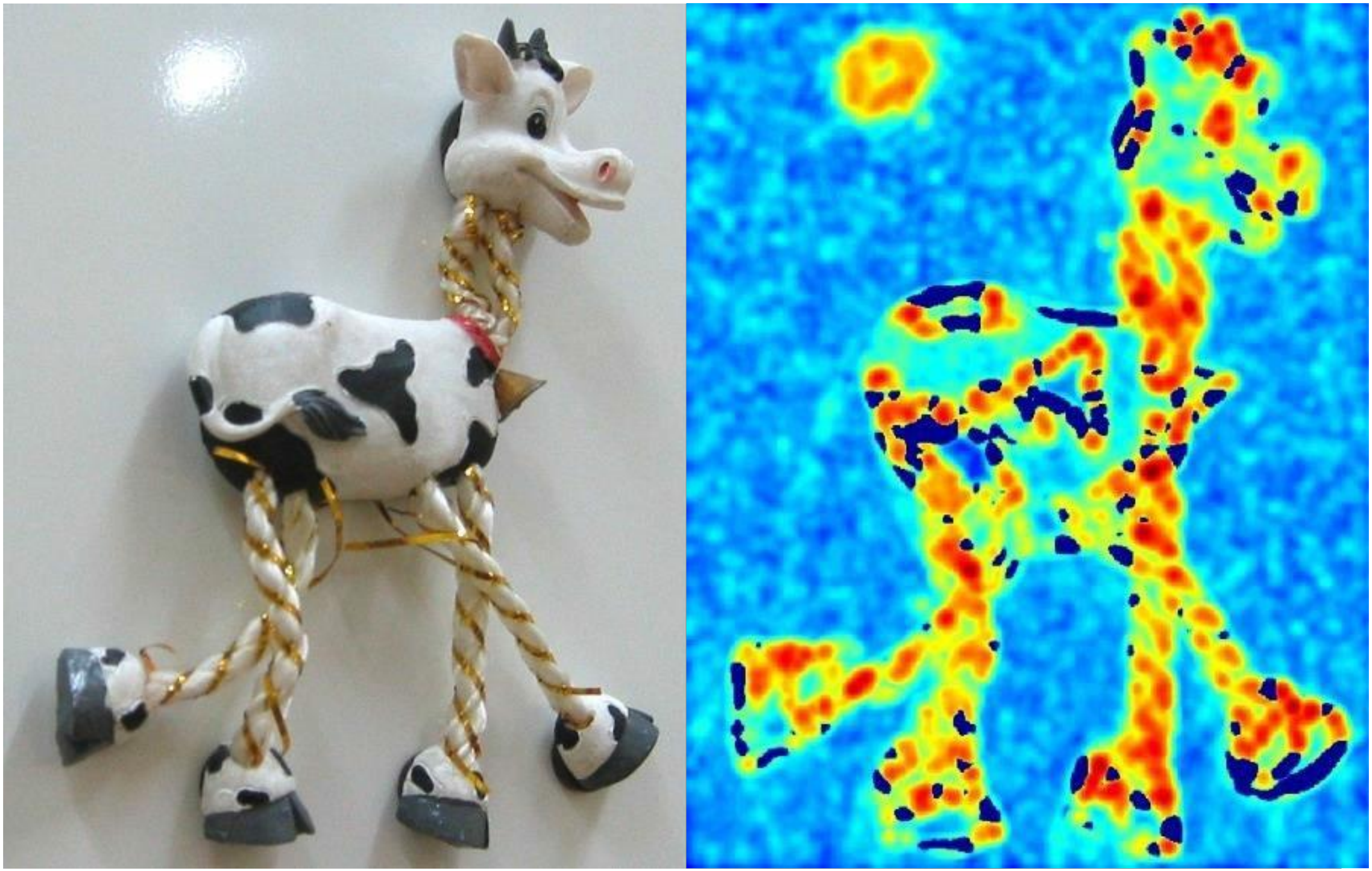
[1] *A. Noble (1989). Descriptions of Image Surfaces (Ph.D.). Department of Engineering Science, Oxford University. p. 45.*

# The Harris operator

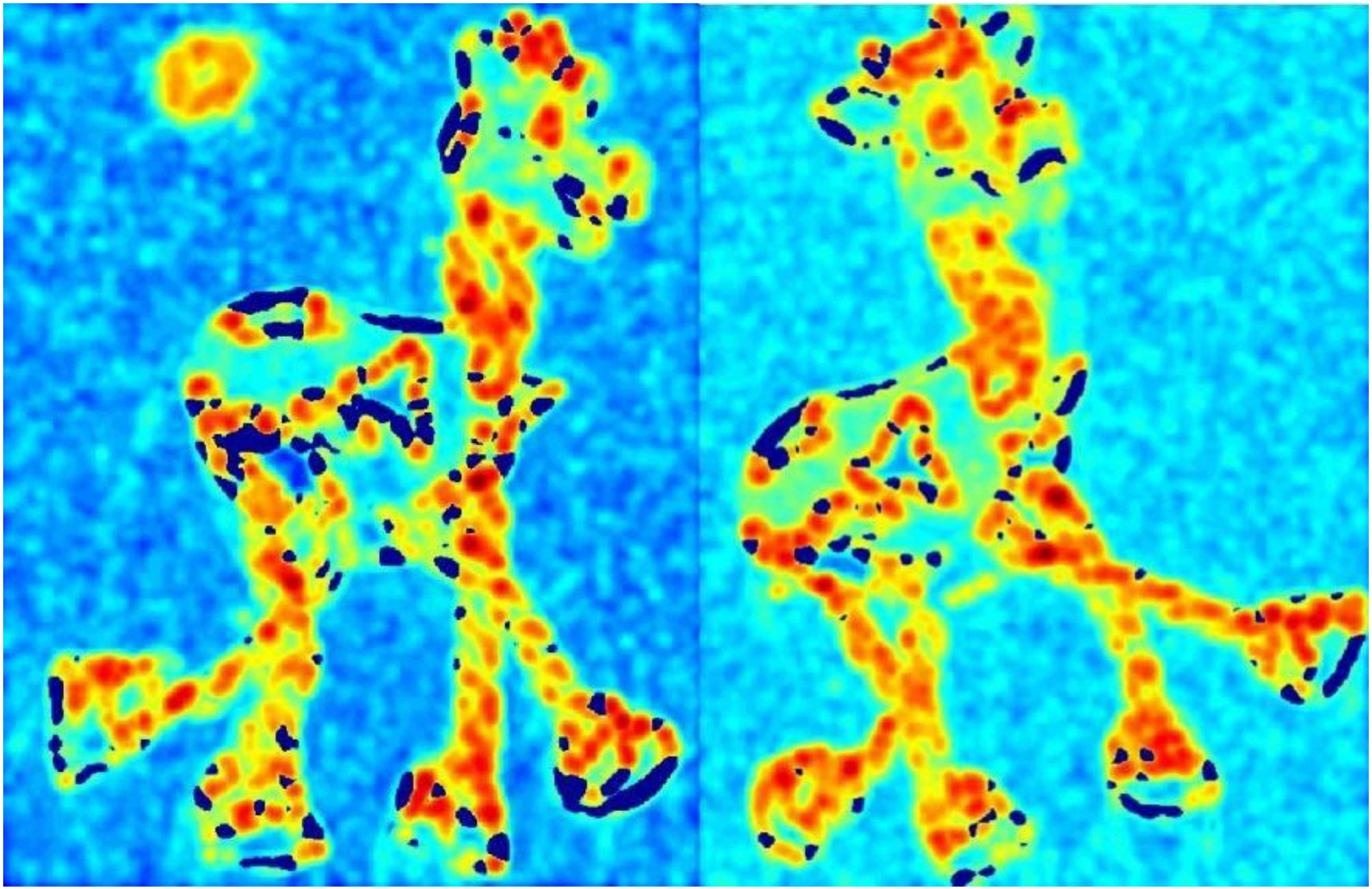


# Harris detector example

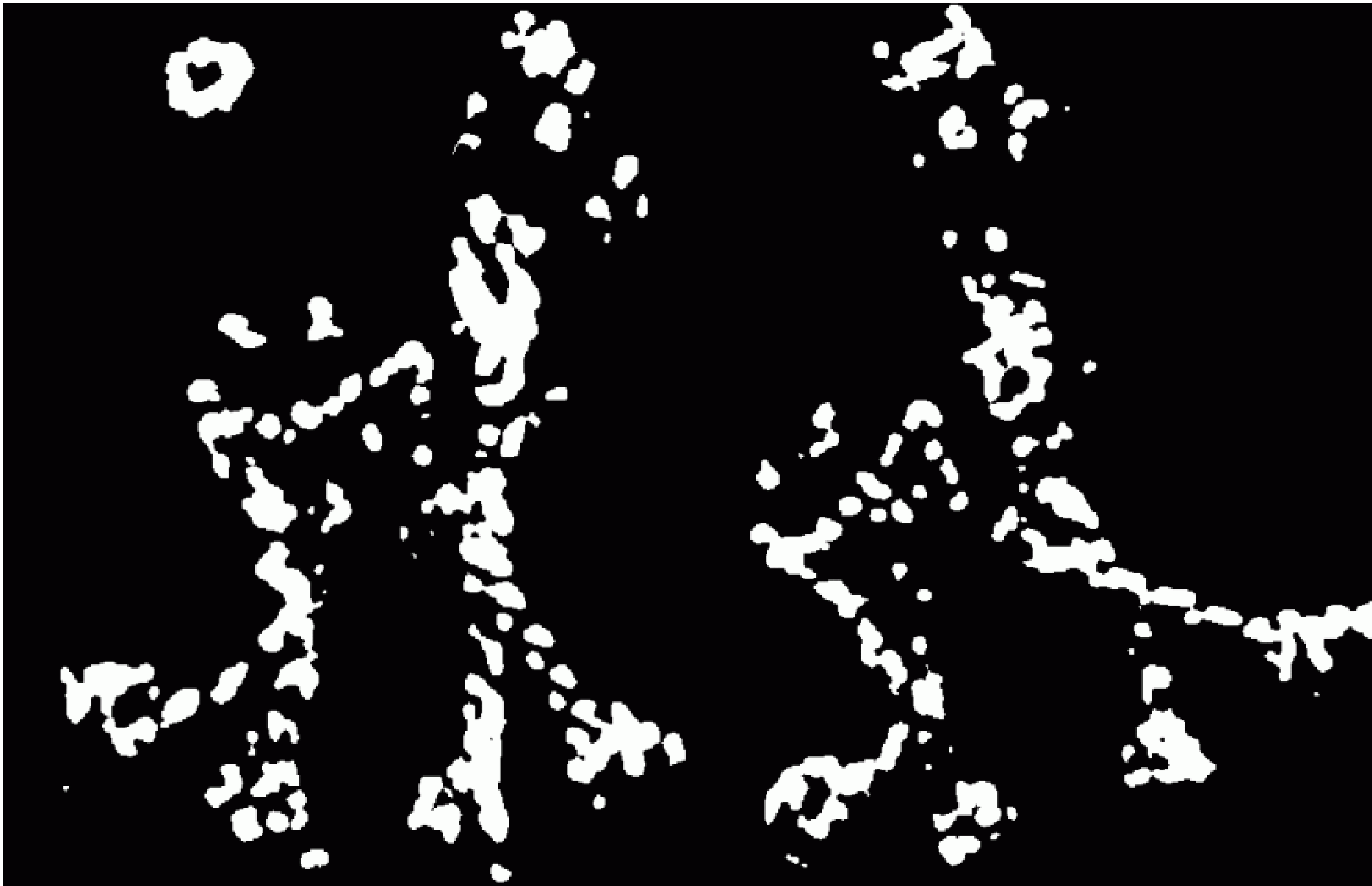
# f value (red high, blue low)



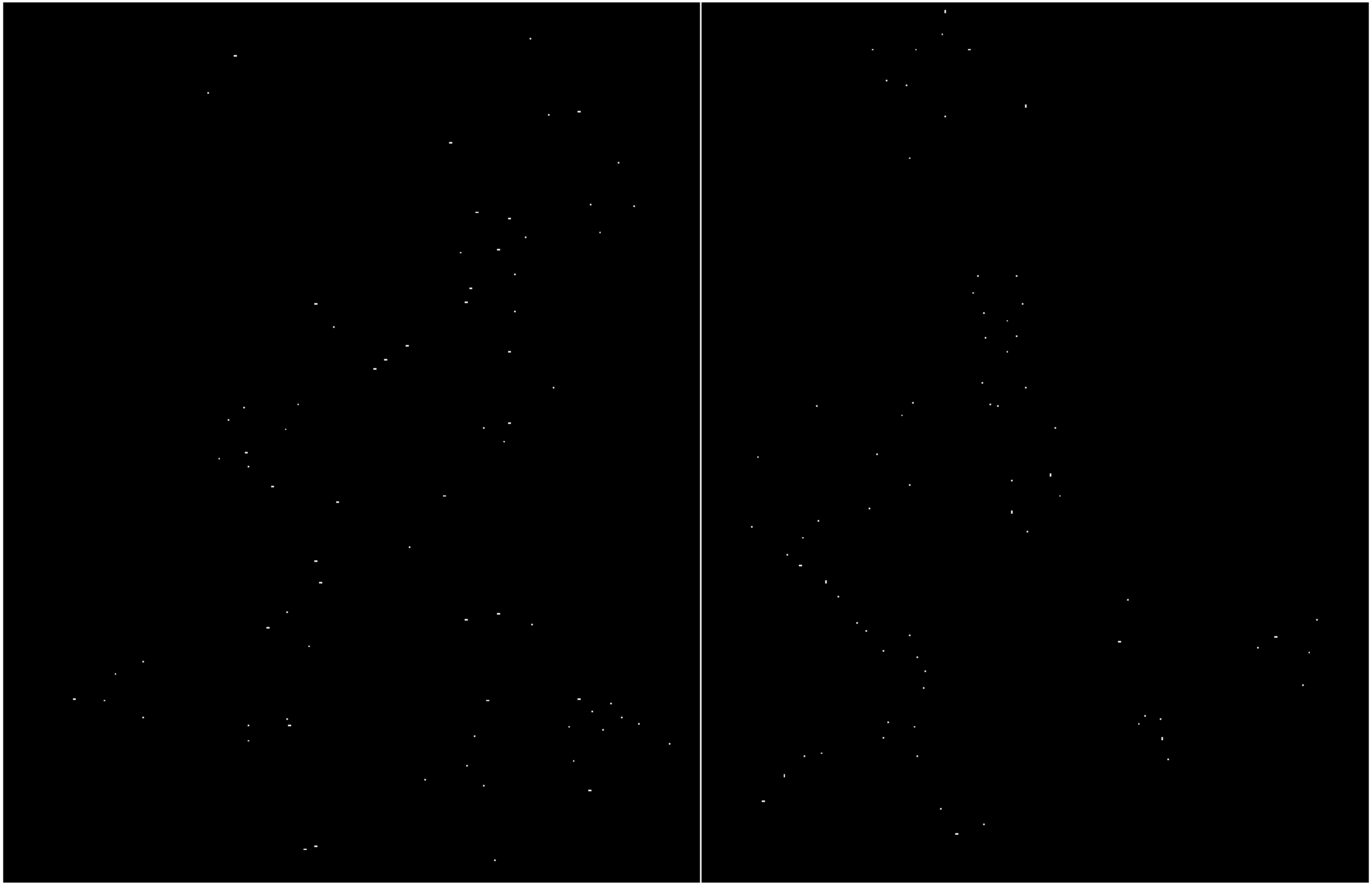
# f value (red high, blue low)



# Threshold (f > value)



# Find local maxima of f



# Harris features (in red)

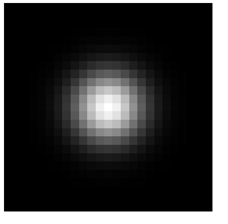
# Weighting the derivatives

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

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

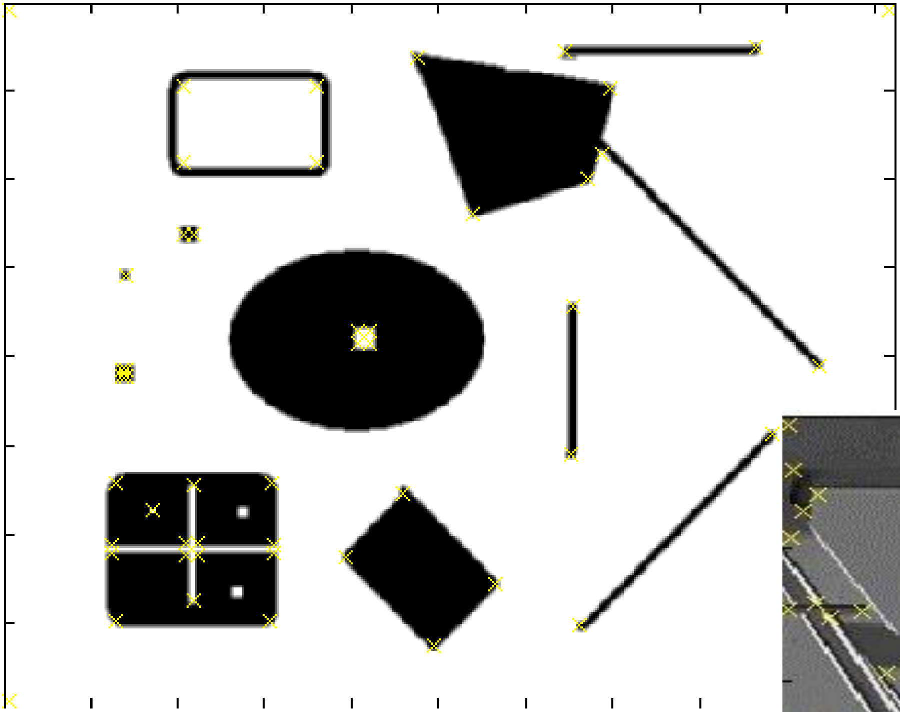
- Instead, we'll *weight* each derivative value based on its distance from the center pixel

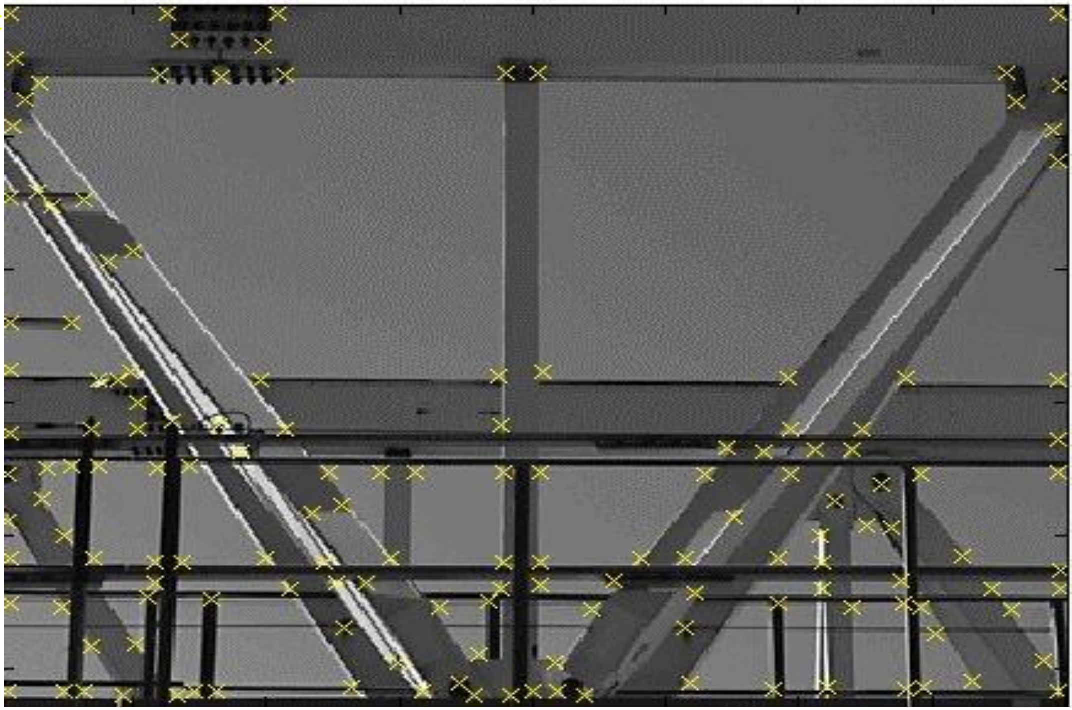
$$H = \sum_{(x,y)\in W} w_{x,y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$



$w_{x,y}$

# Harris Detector – Responses [Harris88]



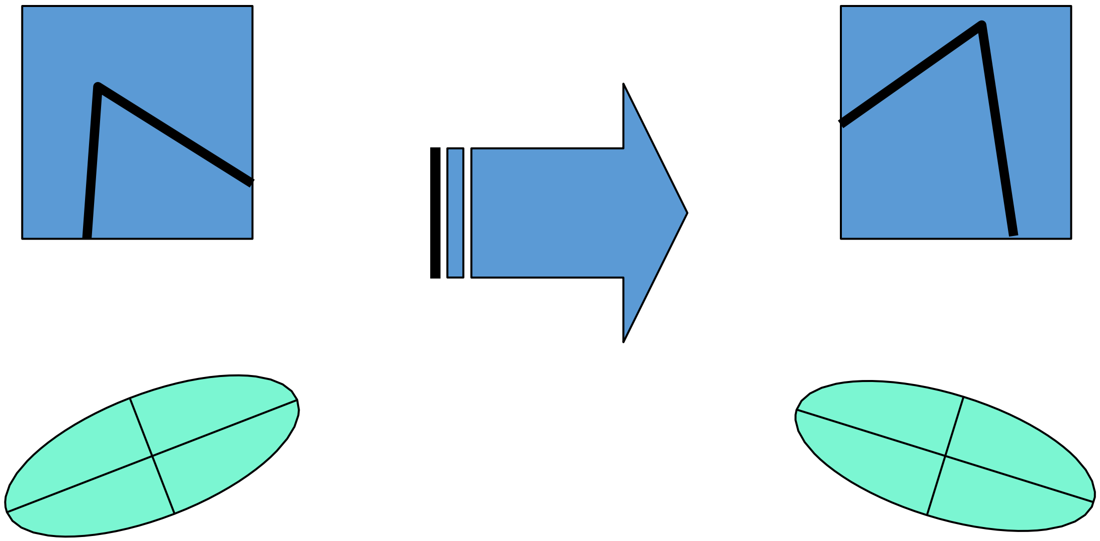


***Effect:* A very precise corner detector.**

# Harris Detector – Responses [Harris88]

# Harris Detector: Invariance Properties

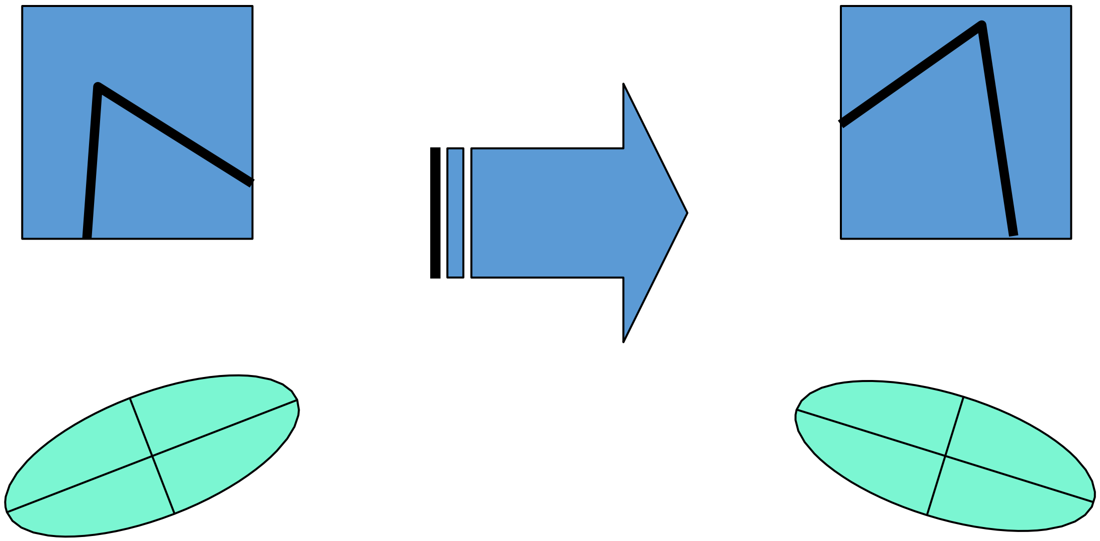
- Rotation



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

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

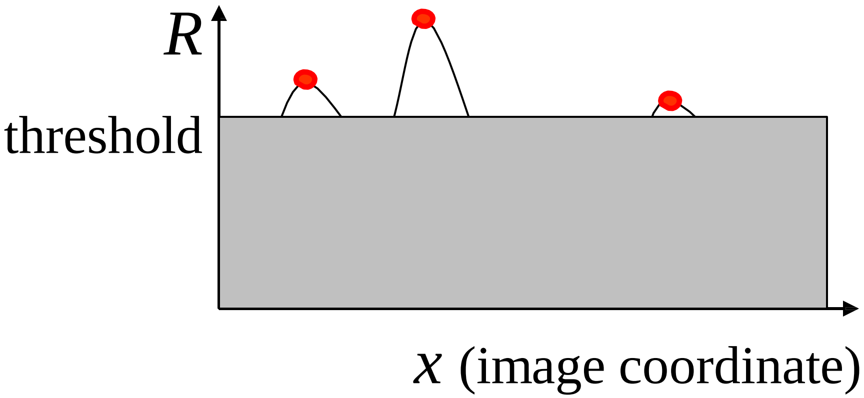
  ✓ Only derivatives are used =>
  invariance to intensity shift $I \rightarrow I + b$

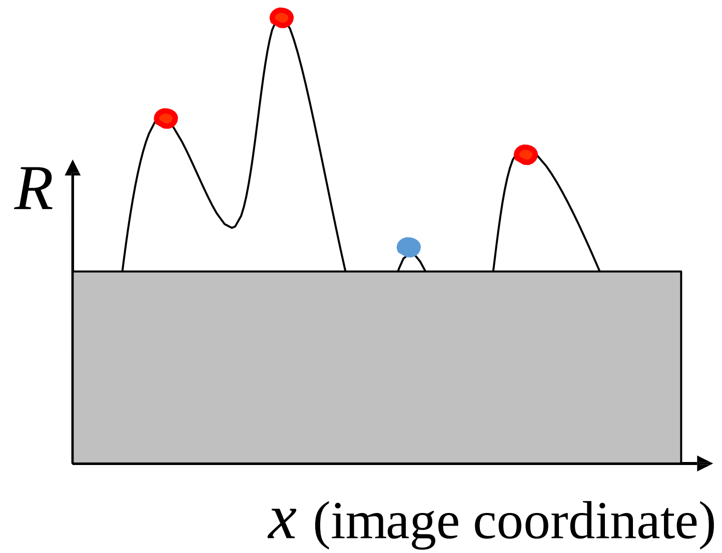
# Harris Detector: Invariance Properties

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

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

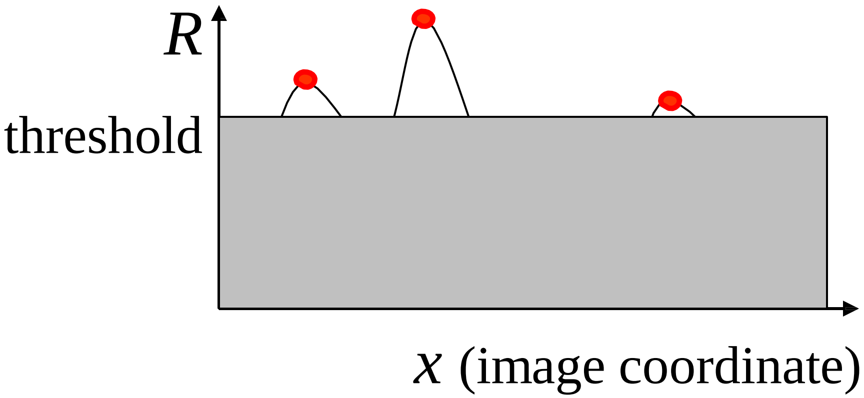
  ✓ Intensity scale: $I \to a\, I$

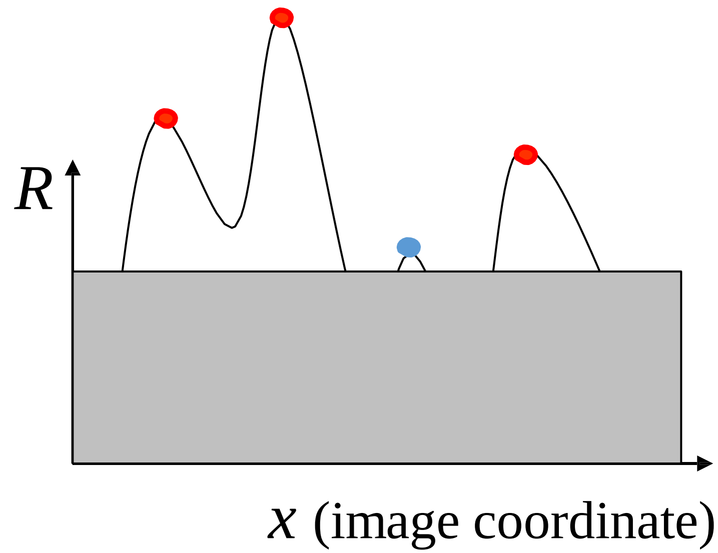


# Harris Detector: Invariance Properties

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

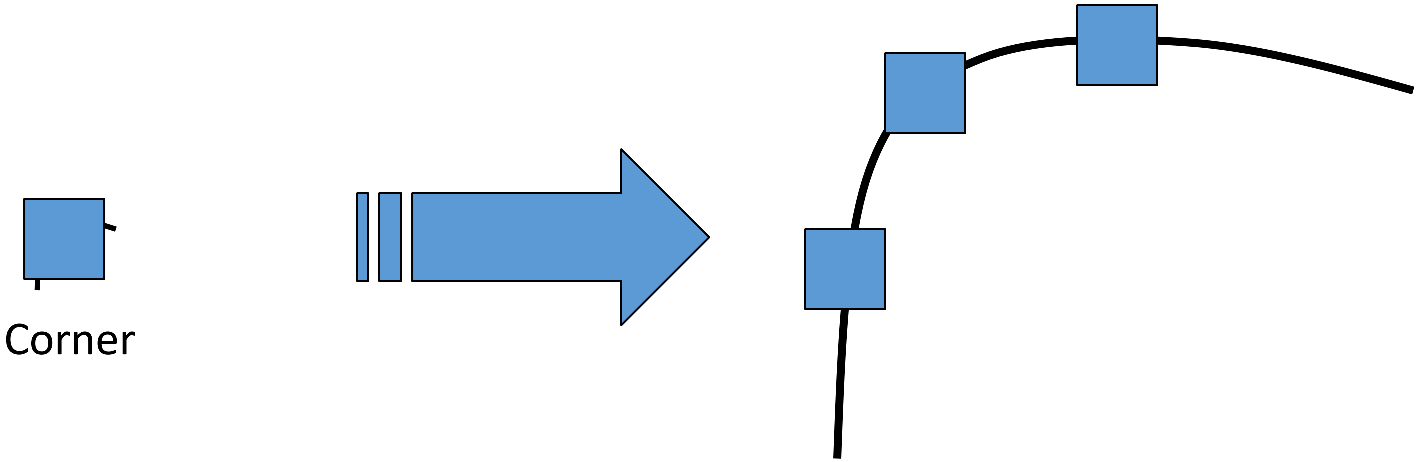



*Partially invariant* to affine intensity change

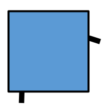
# Harris Detector: Invariance Properties

- Scaling

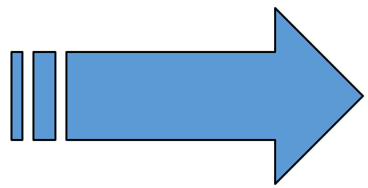


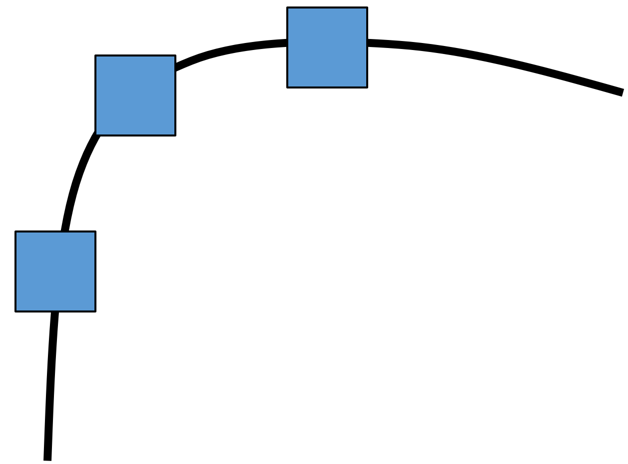
# Harris Detector: Invariance Properties

- Scaling



Corner

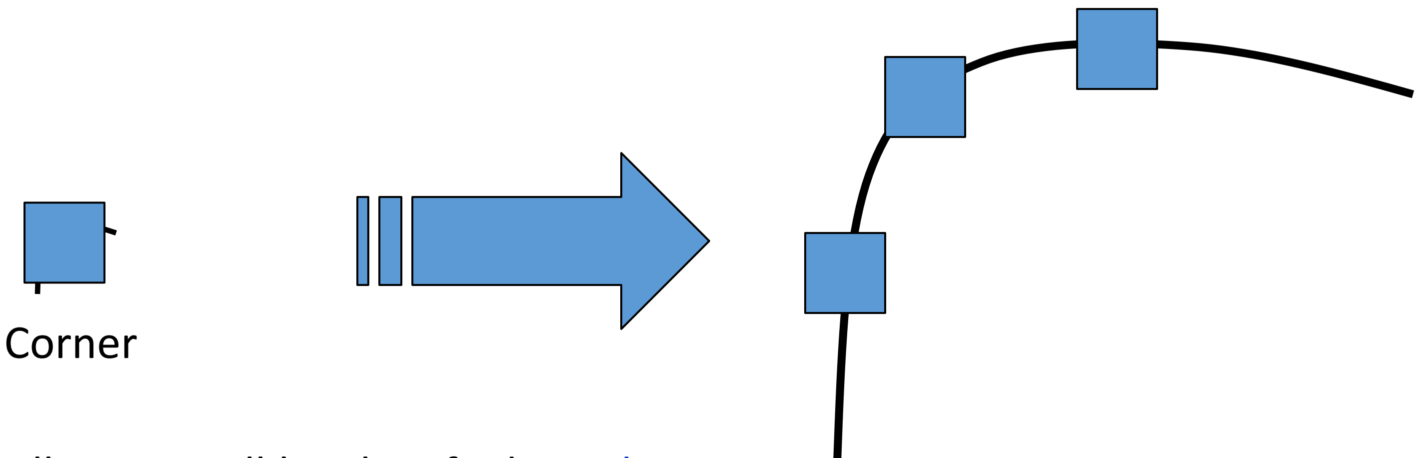




All points will be classified as edges

# Harris Detector: Invariance Properties

- Scaling



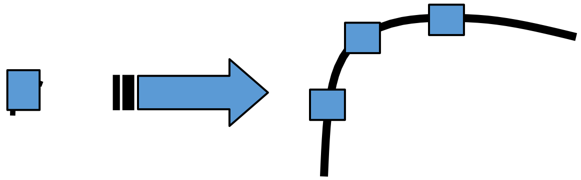

All points will be classified as edges

*Not invariant* to scaling

# Things to remember

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

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



# Acknowledgements


- Thanks to the following researchers for making their teaching/research material online
  - Forsyth
  - Steve Seitz
  - Noah Snavely
  - J.B. Huang
  - Derek Hoiem
  - D. Lowe
  - A. Bobick
  - S. Lazebnik
  - K. Grauman
  - R. Zaleski
  - Leibe

# Thank you

- Next class: Region Detection and Local Descriptors