Image Processing & Vision

Lecture 04: Features

Hak Gu Kim

hakgukim@cau.ac.kr

Immersive Reality & Intelligent Systems Lab (IRIS LAB)

Graduate School of Advanced Imaging Science, Multimedia & Film (GSAIM)

Chung-Ang University (CAU)

27 Mar. 2023

Re-cap: Template Matching

- **Template matching** as (normalized) correlation
- Template matching is **not robust to changes** in
  - 2D spatial scale and 2D orientation (rotation)
  - 3D pose and viewing direction

- **Scaled representations** facilitate:
  - Template matching at multiple scales
  - Efficient search for image-to-image correspondences
  - Image analysis at multiple levels of detail

- A **Gaussian pyramid** reduces artifacts when sub-sampling to coarser scales

Topics

- Features
— Edges
— Corners

***Note:** Many of these slides in this course were adapted from Computer Vision at CMU (16-385) and UBC (CPSC425)

From Template Matching to **Local Feature Detection**

- Global template matching ⟶ **Local feature detection**

- Consider the problem of finding images of an elephant using a template

— An elephant looks different from different viewpoints

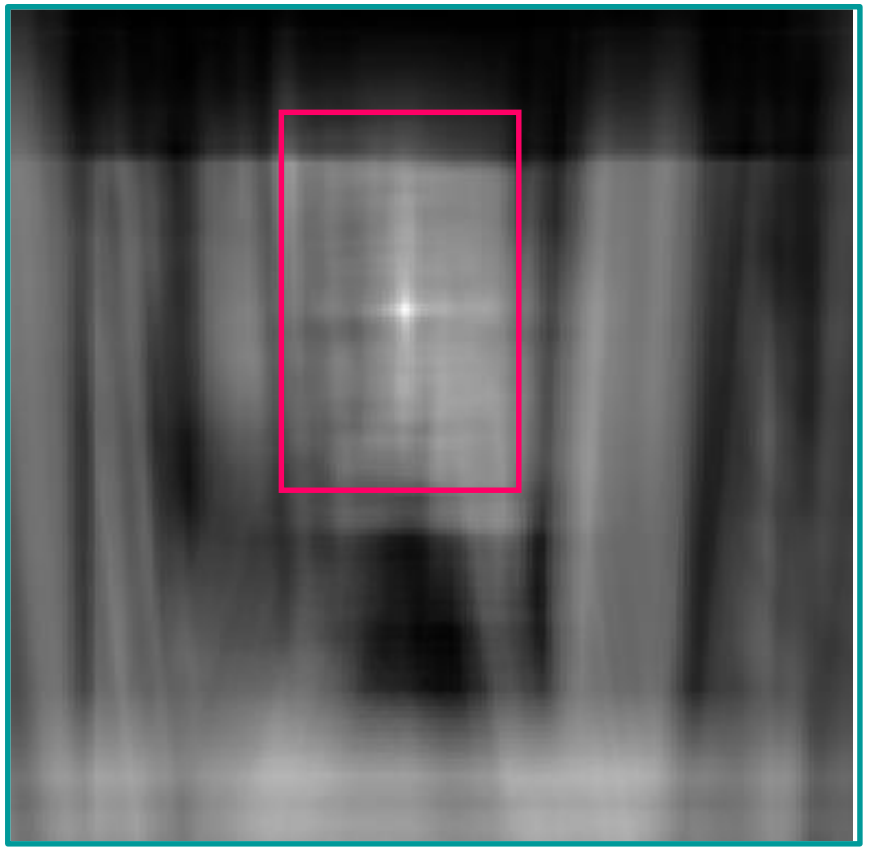
— What happens if parts of an elephant are obscured from view by trees, rocks, other elephants?

From Template Matching to **Local Feature Detection**

- Is recognition really that hard?

Template

Image

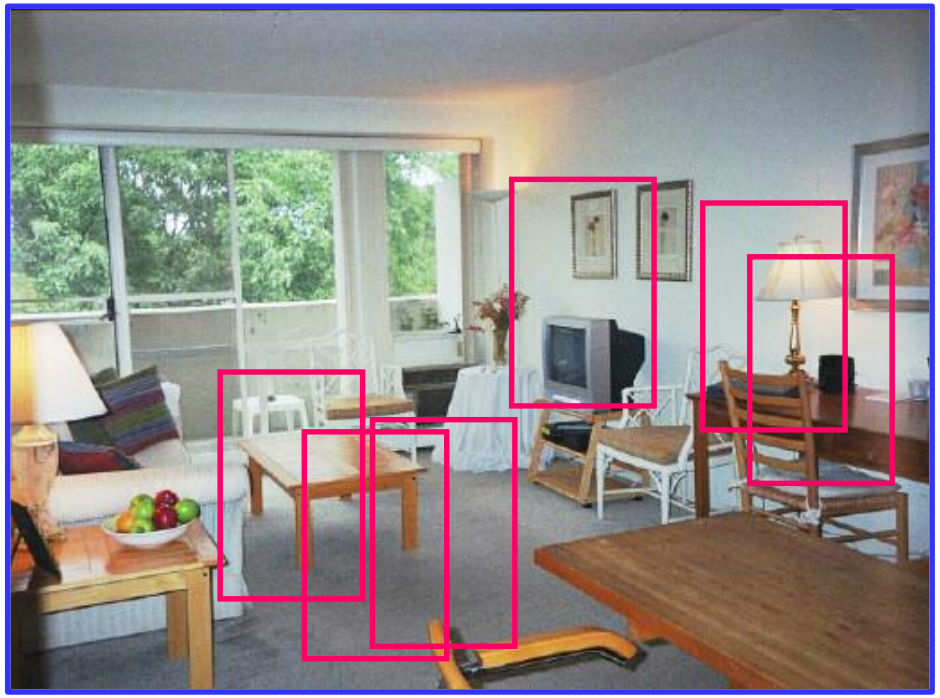


Normalized correlation map

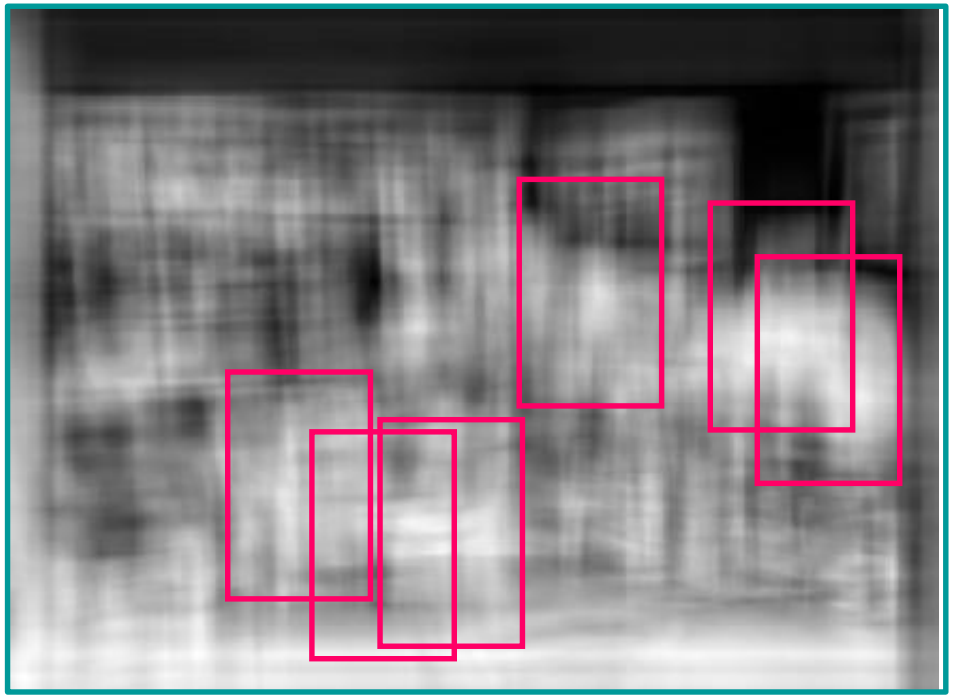
From Template Matching to **Local Feature Detection**

- Is recognition really that hard?

Template



Image



Normalized correlation map

From Template Matching to **Local Feature Detection**

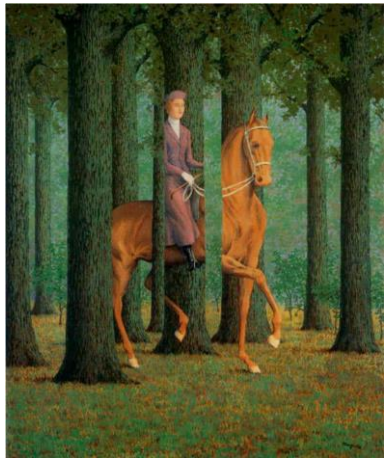
- What makes object recognition hard?

Different viewpoint

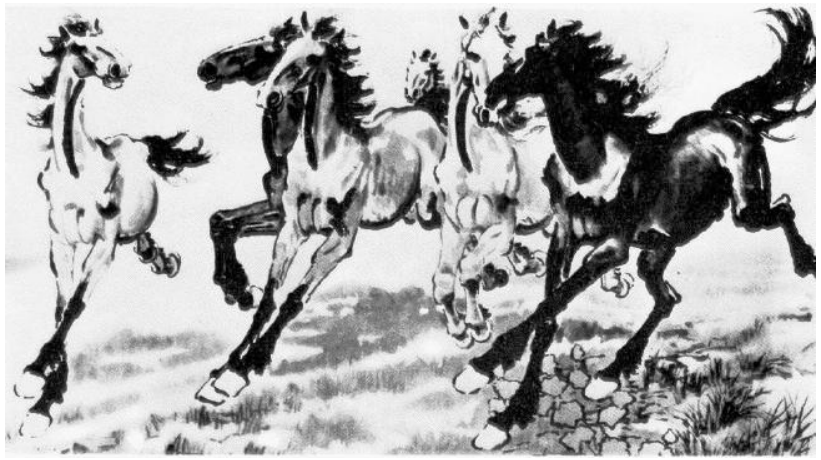
Scale variation

Intra-class variation

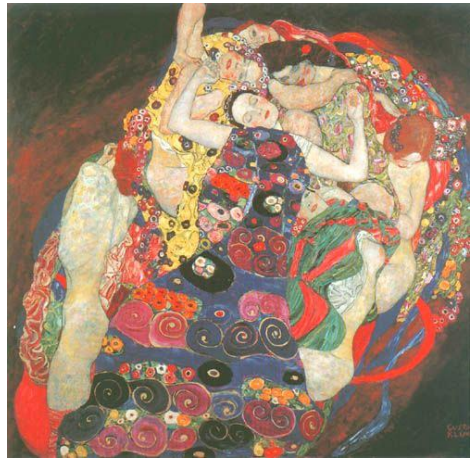
Illumination variation



Occlusion



Deformation



Background clutter

From Template Matching to **Local Feature Detection**

- Global template matching ⟶ **Local feature detection**

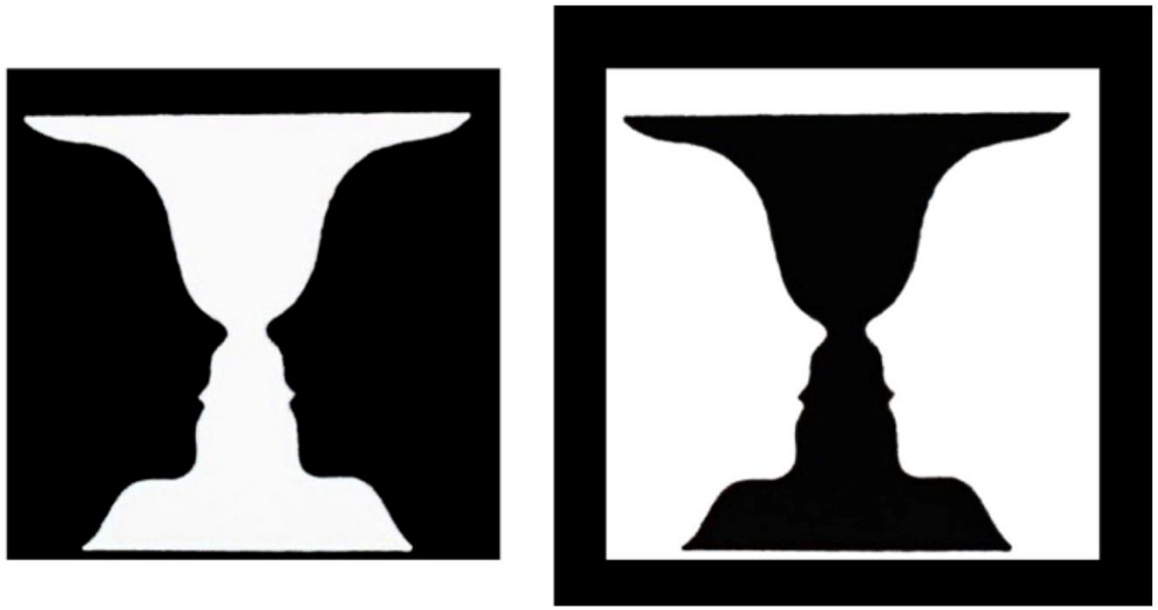
- Move from global template matching to local template matching

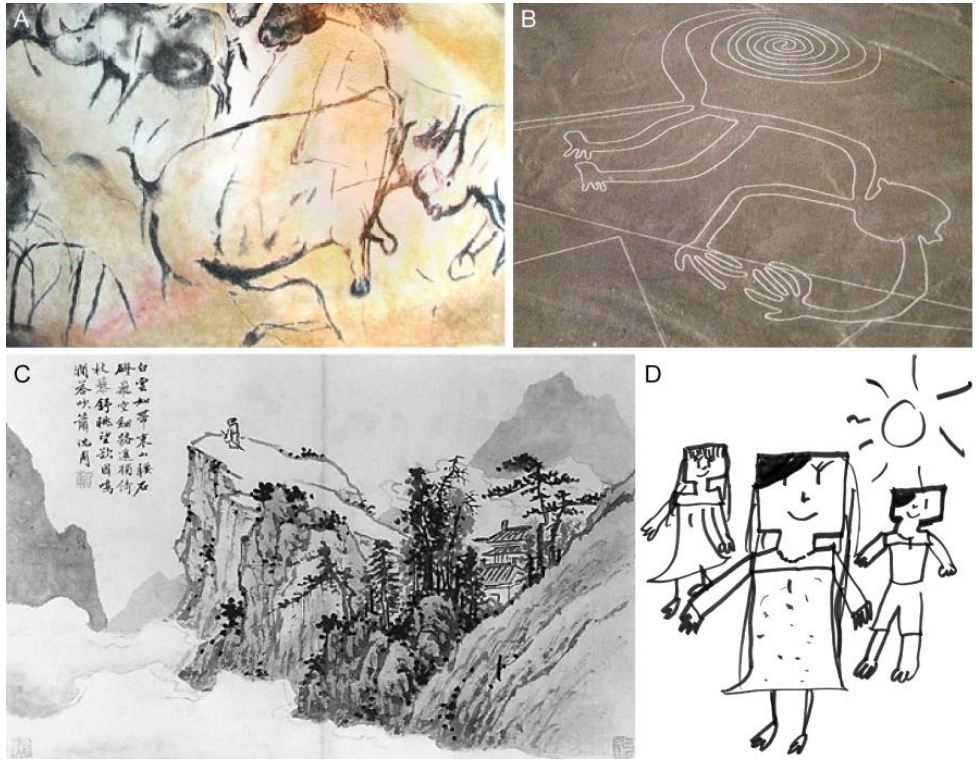
— Local template matching also called **local feature detection**

— Obvious local features to detect are **edges** and **corners**

Edges

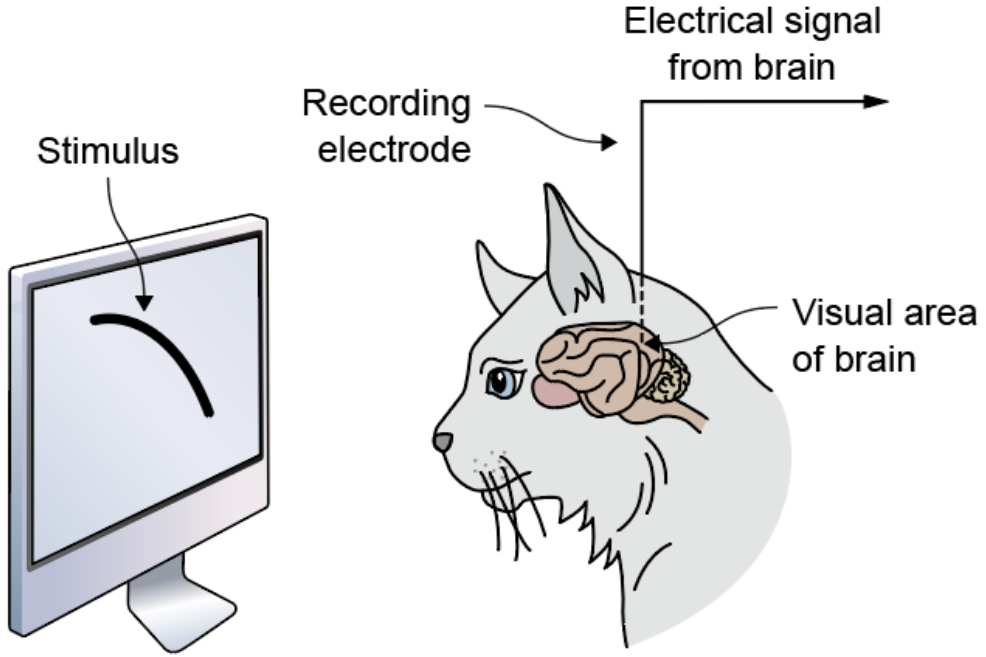
- Edges are significant local changes in the image and are important features for analyzing images
- Edges typically occur on the boundary between two different regions in an image



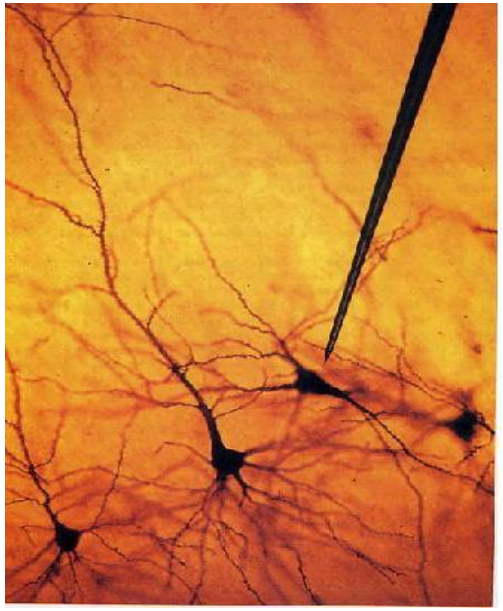


Discoveries about **Visual System and Processing**

- Nobel Prize for Physiology or Medicine in 1981
  - David Hubel and Torsten Wiesel recorded electrical activity from individual neurons in the brains of cats.
  - They discovered that neurons in different positions in the visual cortex are activated if edges with different orientations slide over the retina of the cat's eye.



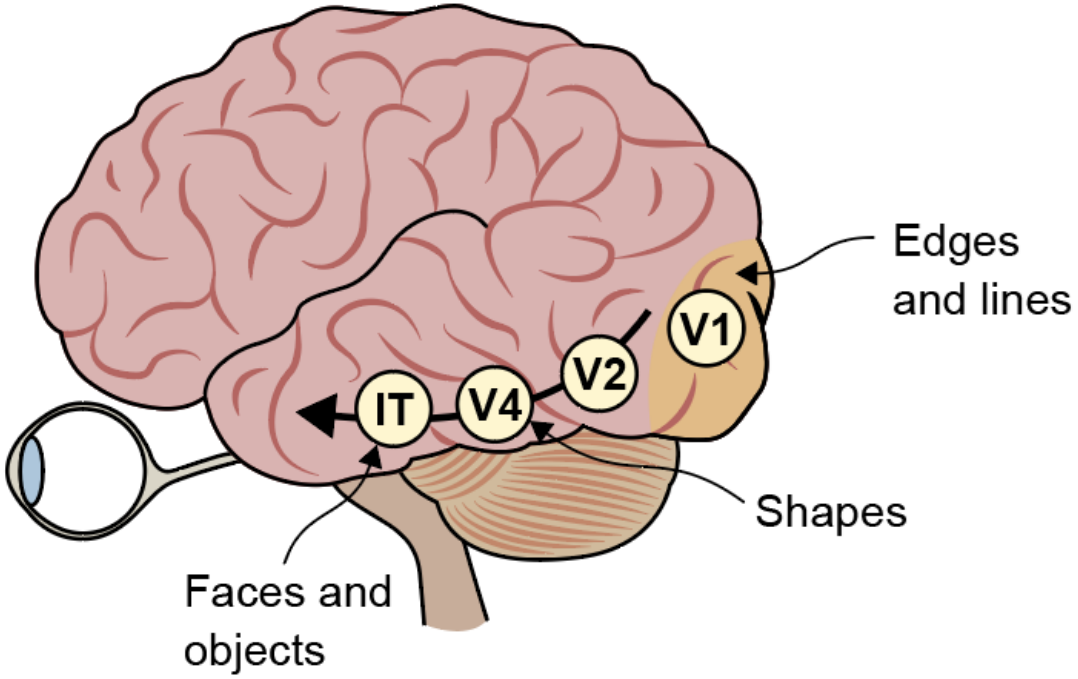
https://freecontent.manning.com/deep-learning-for-image-like-data/

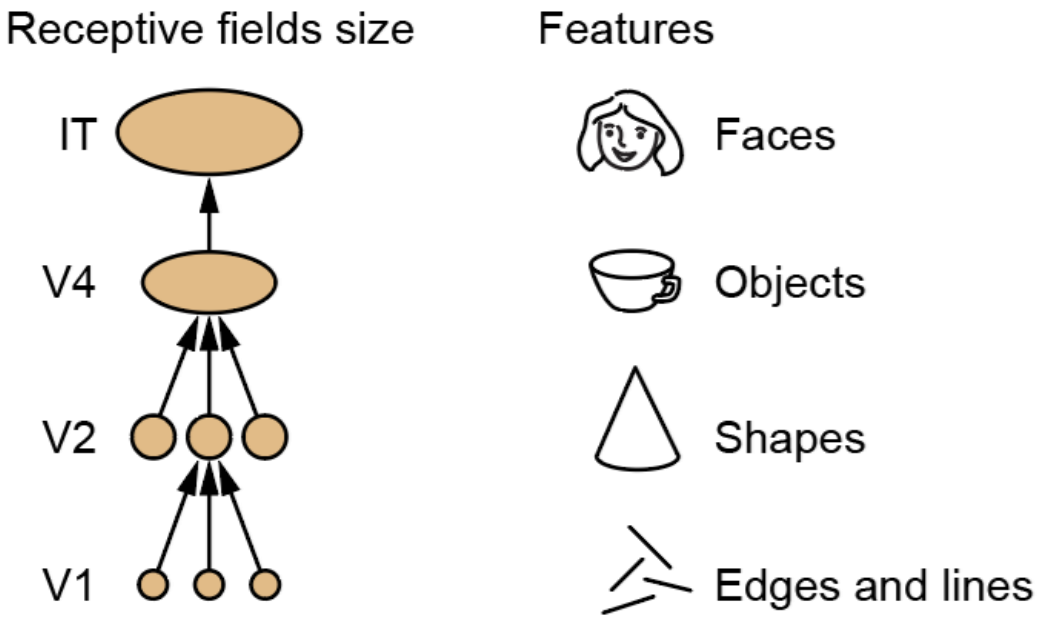


https://www.youtube.com/watch?v=IOHayh06LJ4

Discoveries about **Visual System and Processing**

- Organization of the **Visual Cortex**
  - It is widely known that in V1 region, all neurons respond to rather simple forms of stimuli on different areas of the retina.
  - It is also known that neurons in other regions of the brain (called V2, V4, and IT) respond to increasingly complex visual stimuli like, for example, a whole face.

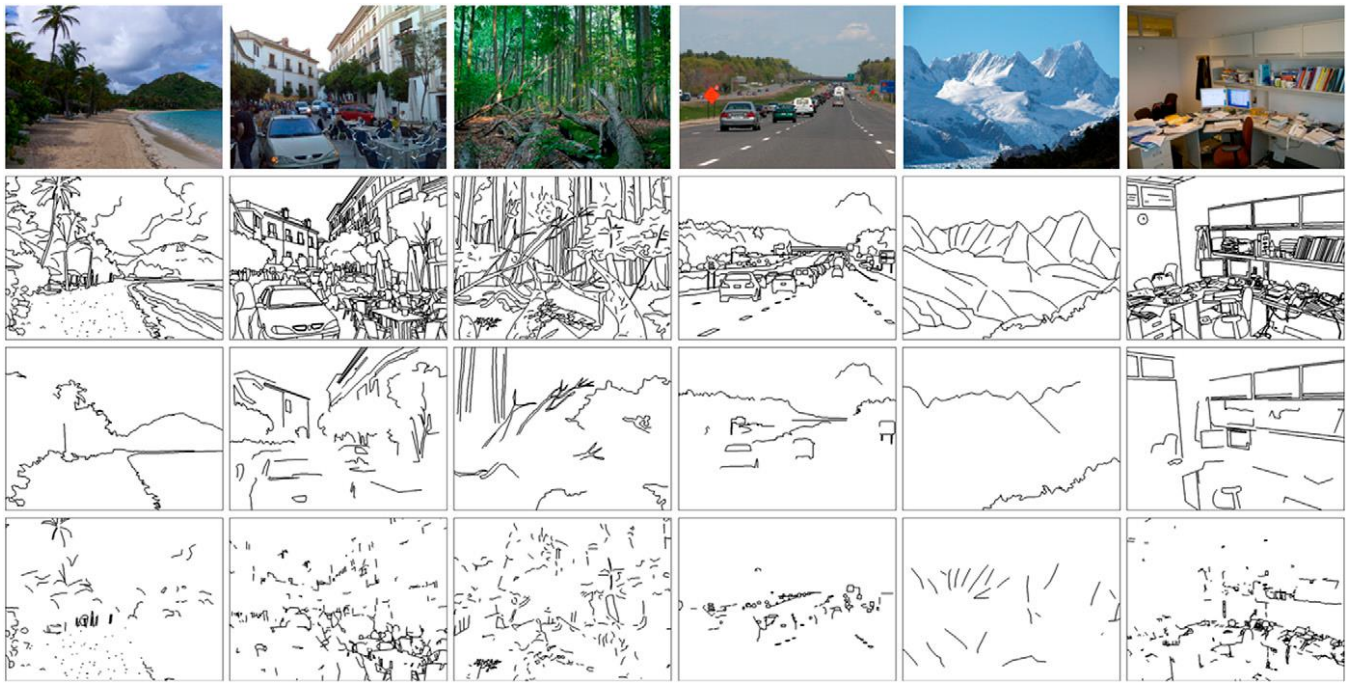


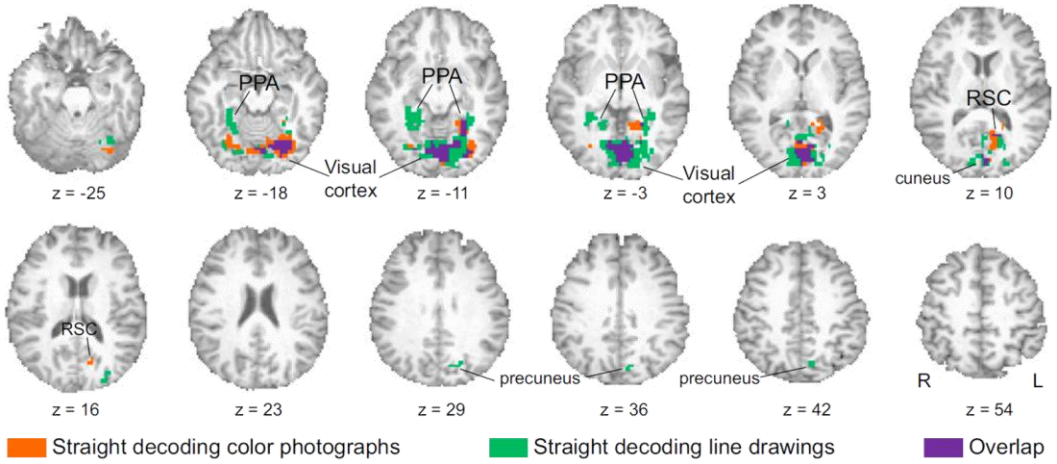


https://freecontent.manning.com/deep-learning-for-image-like-data/

Discoveries about **Visual System and Processing**

- **Color Photographs vs. Line Drawings**
  - To decode scene category from fMRI data for line drawings just as well as from activity for color photographs





PPA decoder prediction

	bea	city	for	hwy	mnt	off
beaches	**.25**	.09	.09	.21	.20	.16
city streets	.14	**.27**	.05	.18	.09	.27
forests	.17	.14	**.33**	.14	.14	.08
highways	.17	.14	.09	**.36**	.07	.17
mountains	.20	.09	.17	.19	**.31**	.04
offices	.11	.13	.05	.23	.07	**.40**

Color photographs

	bea	city	for	hwy	mnt	off
	.16	.14	.22	.23	.13	.12
	.13	**.28**	.10	.14	.04	.31
	.15	.10	**.31**	.14	.25	.05
	.25	.21	.09	**.25**	.06	.14
	.31	.06	.15	.04	**.39**	.05
	.08	.28	.03	.22	.05	**.35**

Line drawings

Simple line drawings suffice for functional MRI decoding of natural scene categories, **PNAS, 2011**

Estimating **Derivatives**

- For a 2D continuous function, $f(x, y)$

$$\frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{f(x+h, y) - f(x, y)}{h}$$

  - Differentiation is linear and shift invariant, and can be implemented as a convolution

- A discrete approximation:
  - Δx=1: The smallest unit

$$\frac{\partial f}{\partial x} \approx \frac{f(x+1, y) - f(x, y)}{\Delta x} \longrightarrow \frac{df}{dx} = f(x+1) - f(x) = f'(x)$$

-1	1

Estimating **Derivatives**

- A discrete approximation:

$$\frac{\partial f}{\partial x} \approx \frac{f(x+1,y) - f(x,y)}{\Delta x} \longrightarrow \frac{df}{dx} = f(x+1) - f(x) = f'(x)$$

- **Forward** difference

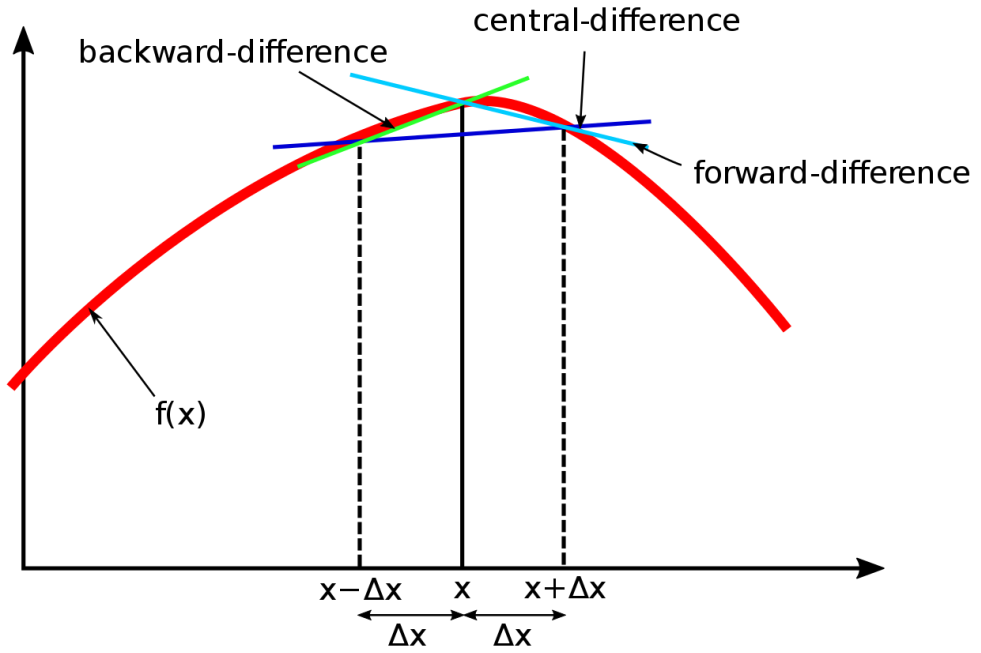
$$\frac{df}{dx} = f(x) - f(x+1)$$

1	-1

- **Backward** difference

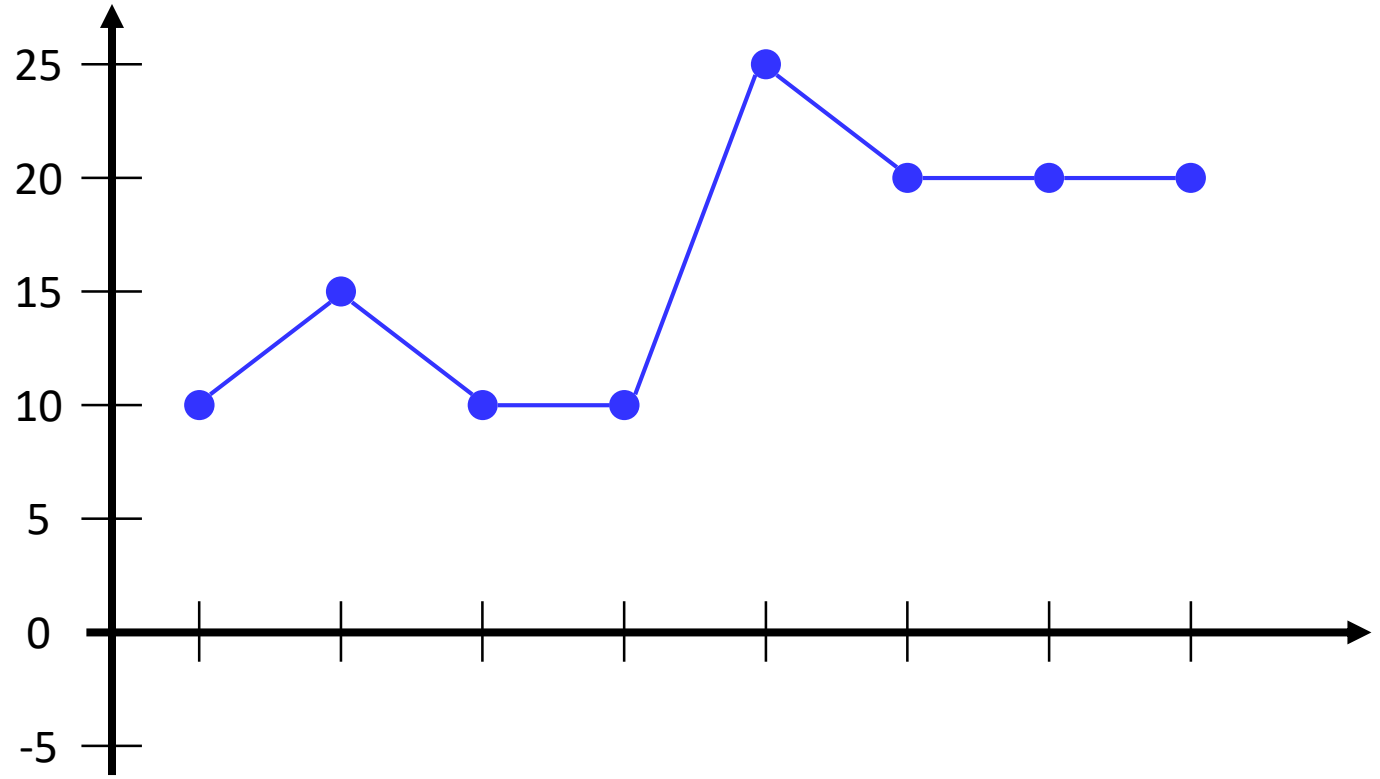
$$\frac{df}{dx} = f(x) - f(x-1)$$

-1	1



https://en.wikipedia.org/wiki/Finite_difference

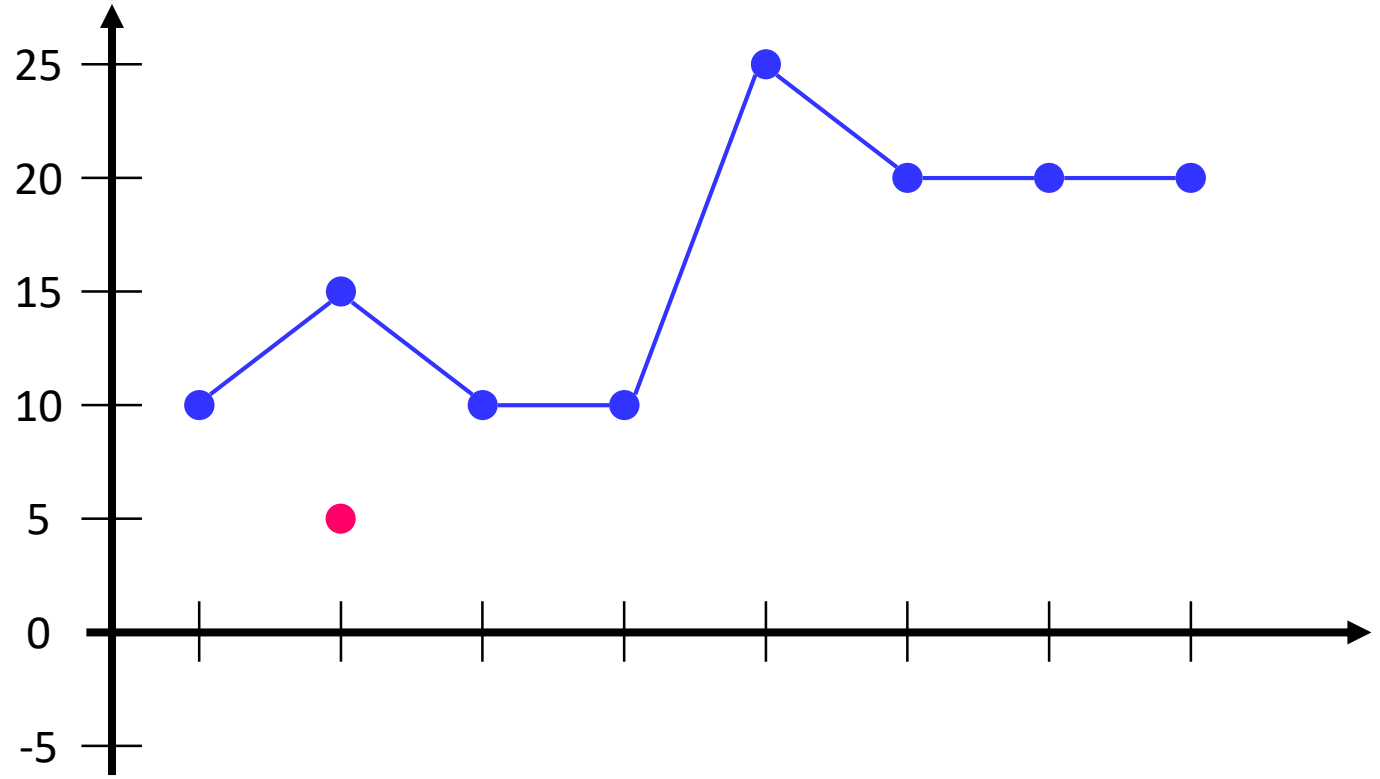
Example: Estimating Derivatives



-1	1

n	1	2	3	4	5	6	7	8
$f(x)$	10	15	10	10	25	20	20	20
$f'(x)$	-							

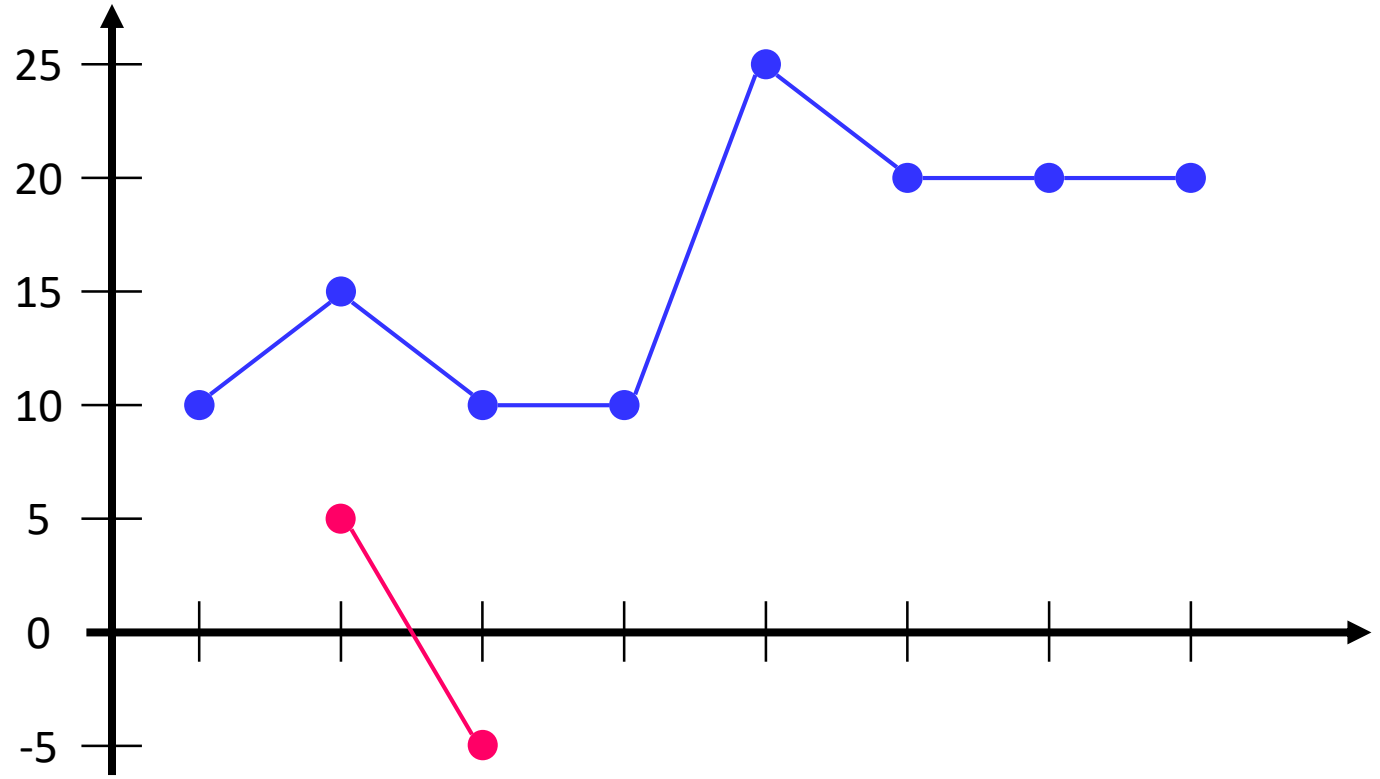
Example: Estimating Derivatives



-1	1

n	1	2	3	4	5	6	7	8
$f(x)$	10	15	10	10	25	20	20	20
$f'(x)$	-	5						

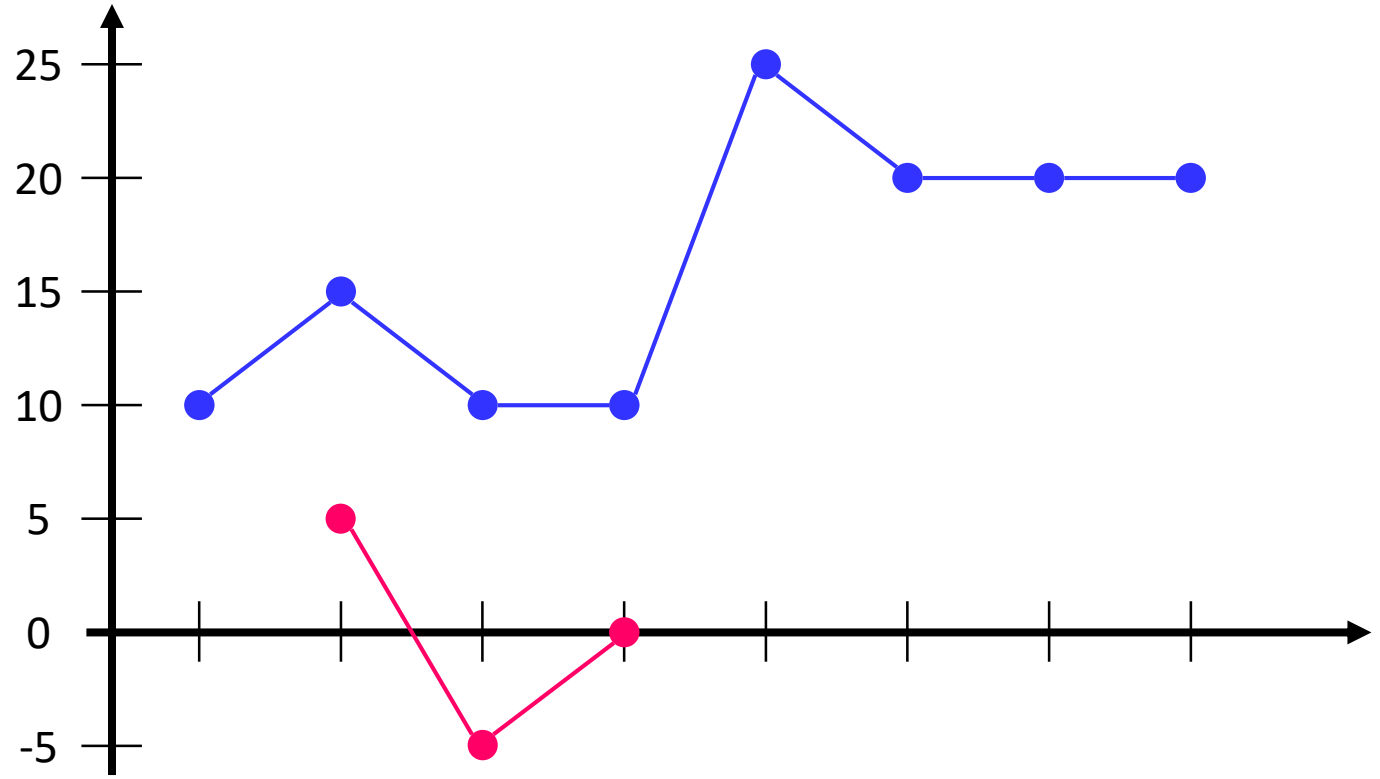
Example: Estimating Derivatives



-1	1

n	1	2	3	4	5	6	7	8
$f(x)$	10	15	10	10	25	20	20	20
$f'(x)$	-	5	-5					

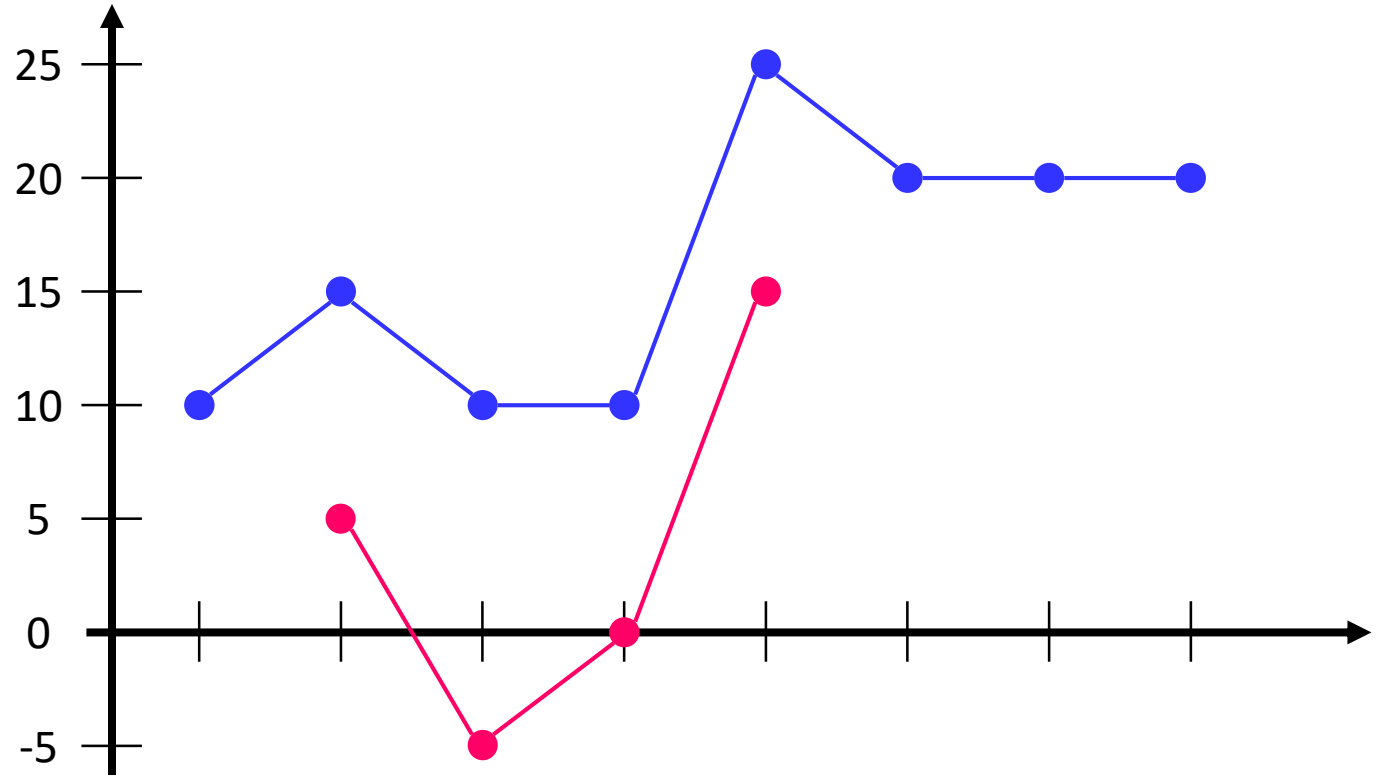
Example: Estimating Derivatives



-1	1

n	1	2	3	4	5	6	7	8
$f(x)$	10	15	10	10	25	20	20	20
$f'(x)$	-	5	-5	0				

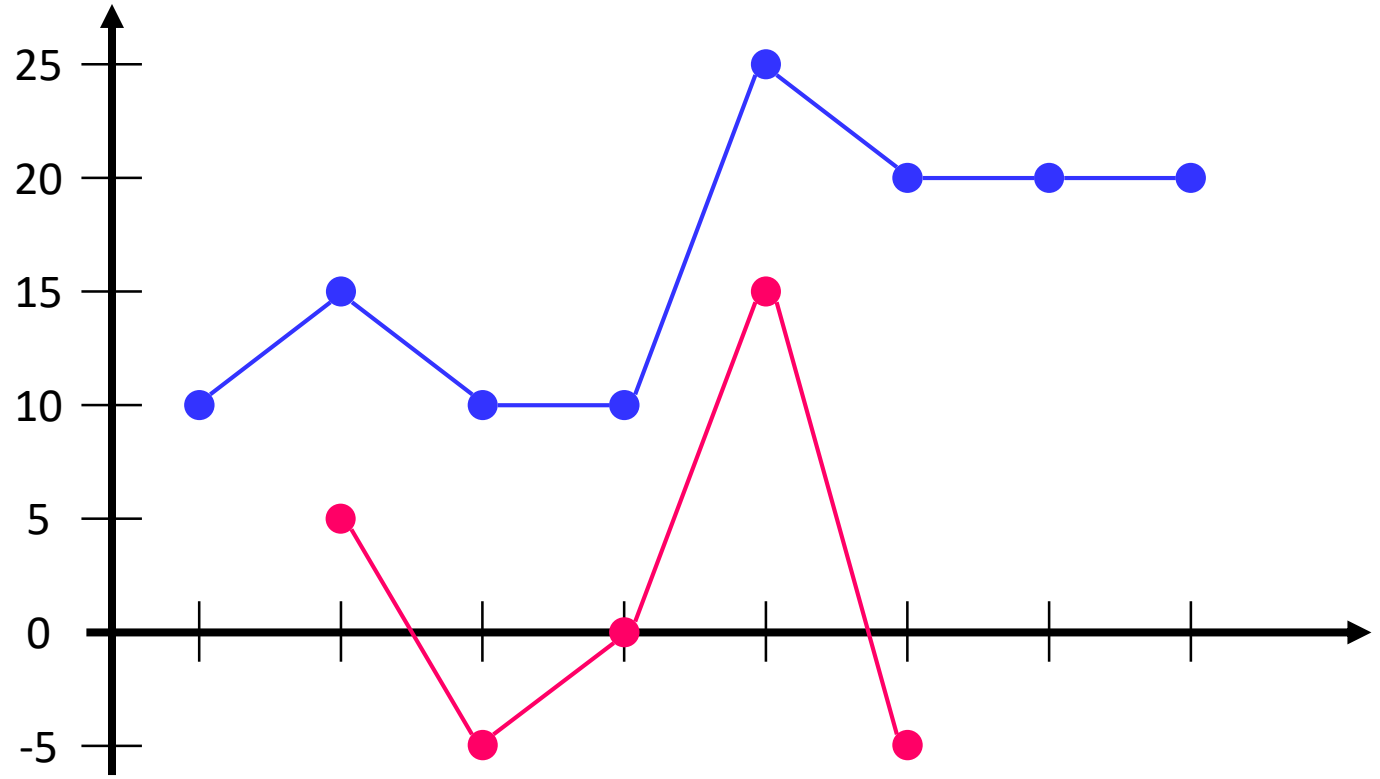
Example: Estimating Derivatives



-1	1

n	1	2	3	4	5	6	7	8
$f(x)$	10	15	10	10	25	20	20	20
$f'(x)$	-	5	-5	0	15			

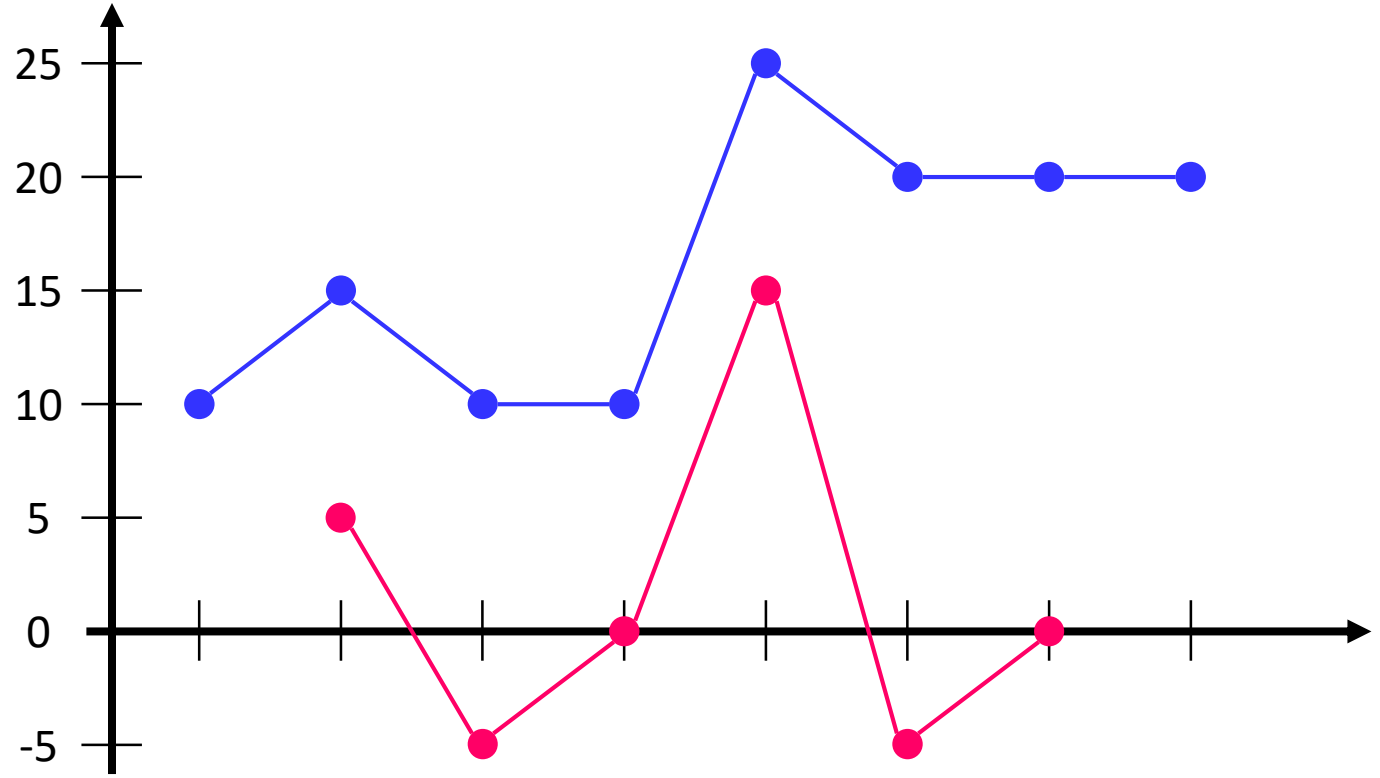
Example: Estimating Derivatives



-1	1

n	1	2	3	4	5	6	7	8
$f(x)$	10	15	10	10	25	20	20	20
$f'(x)$	-	5	-5	0	15	-5		

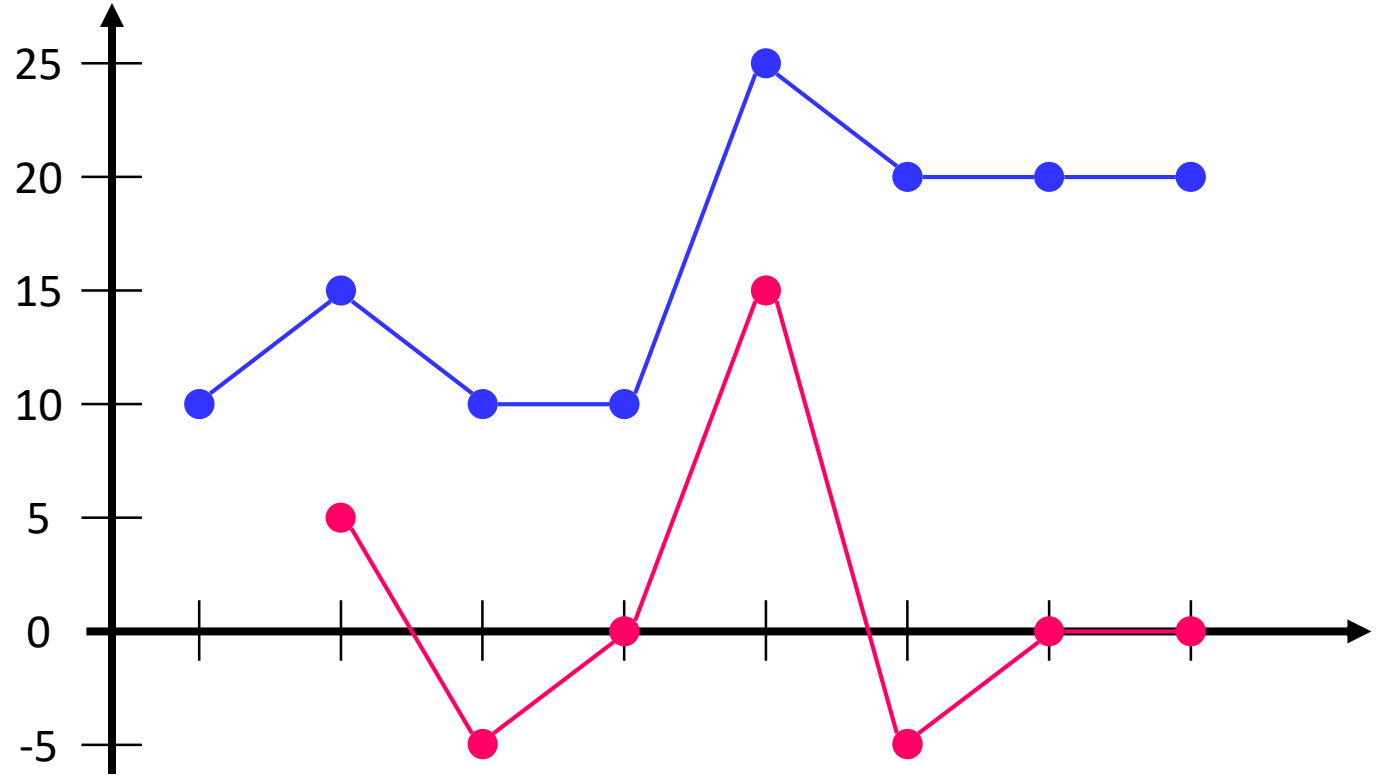
Example: Estimating Derivatives



-1	1

n	1	2	3	4	5	6	7	8
$f(x)$	10	15	10	10	25	20	20	20
$f'(x)$	-	5	-5	0	15	-5	0	

Example: Estimating Derivatives

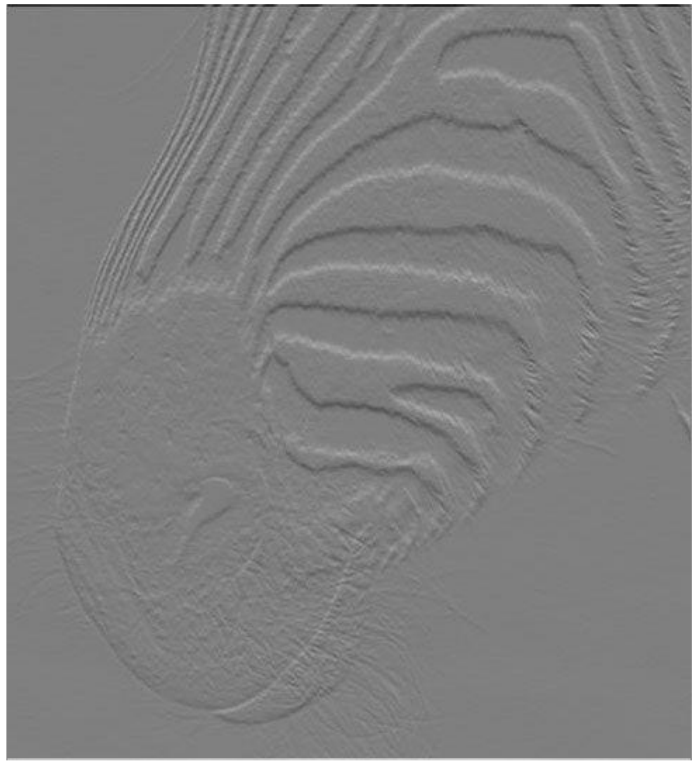


-1	1

n	1	2	3	4	5	6	7	8
$f(x)$	10	15	10	10	25	20	20	20
$f'(x)$	-	5	-5	0	15	-5	0	0

Example: Estimating Derivatives

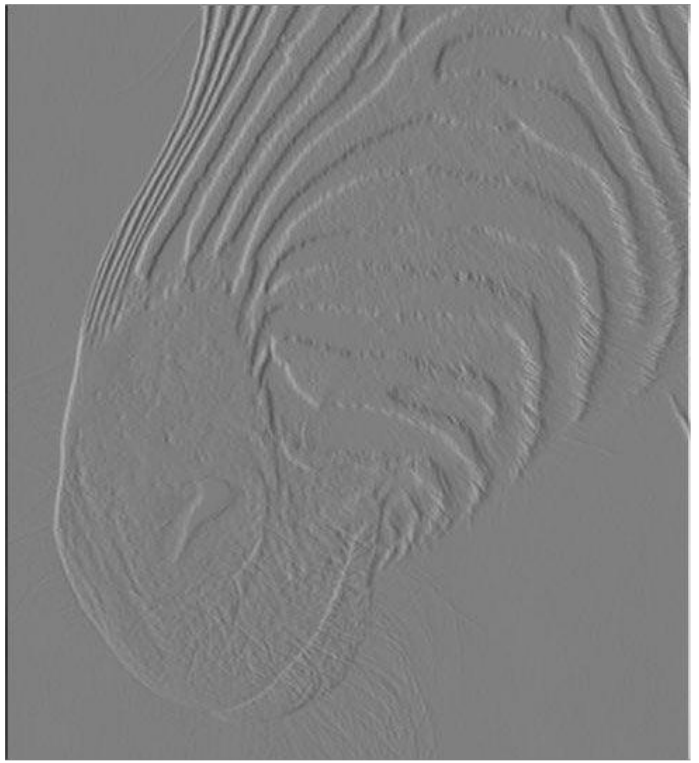
- Derivative **in Y direction: Vertical**



Derivation in y direction

- Derivative **in X direction: Horizontal**

Original image



Derivation in x direction

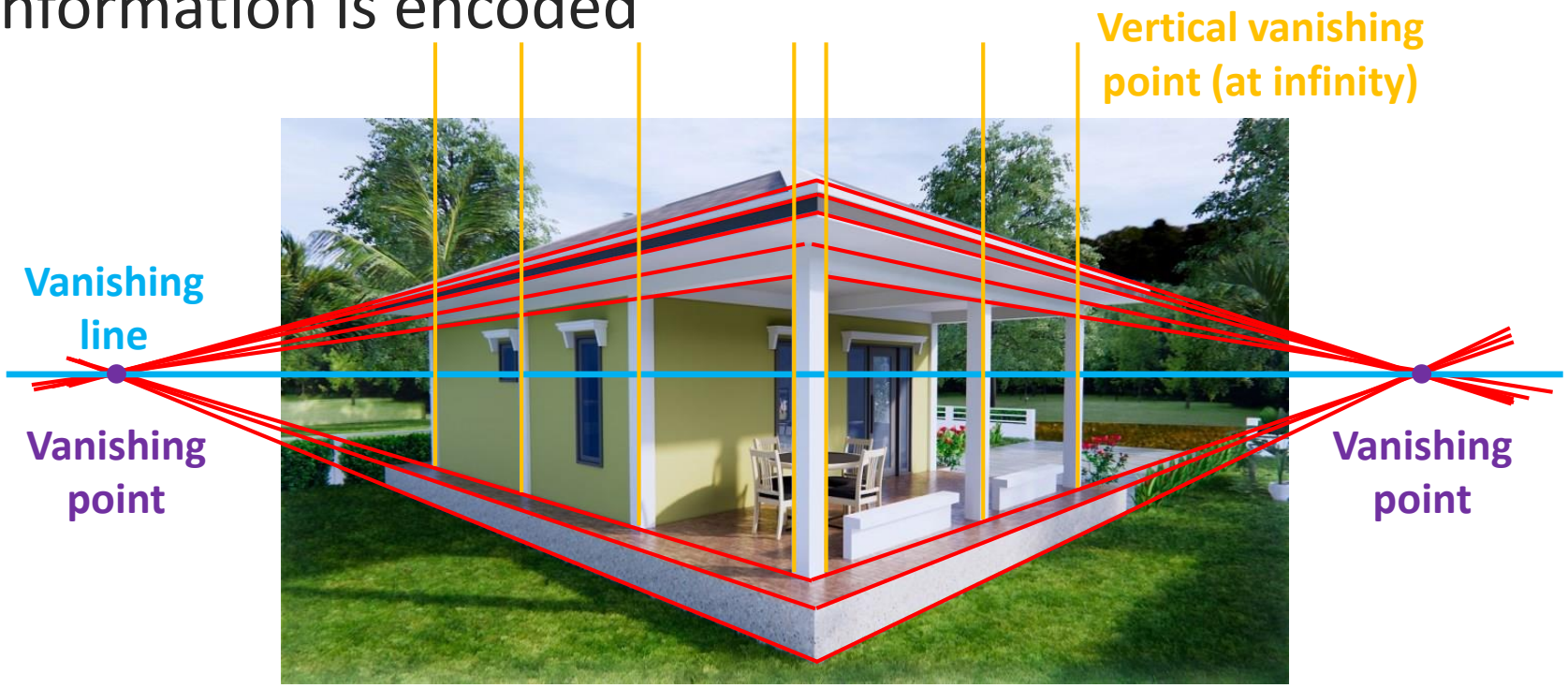
Edge Detection

- **Goal**

— Identify sudden changes in image intensity

— It is where most shape information is encoded

Information extraction
& Object recognition



Geometry and viewpoint recover
& 3D reconstruction

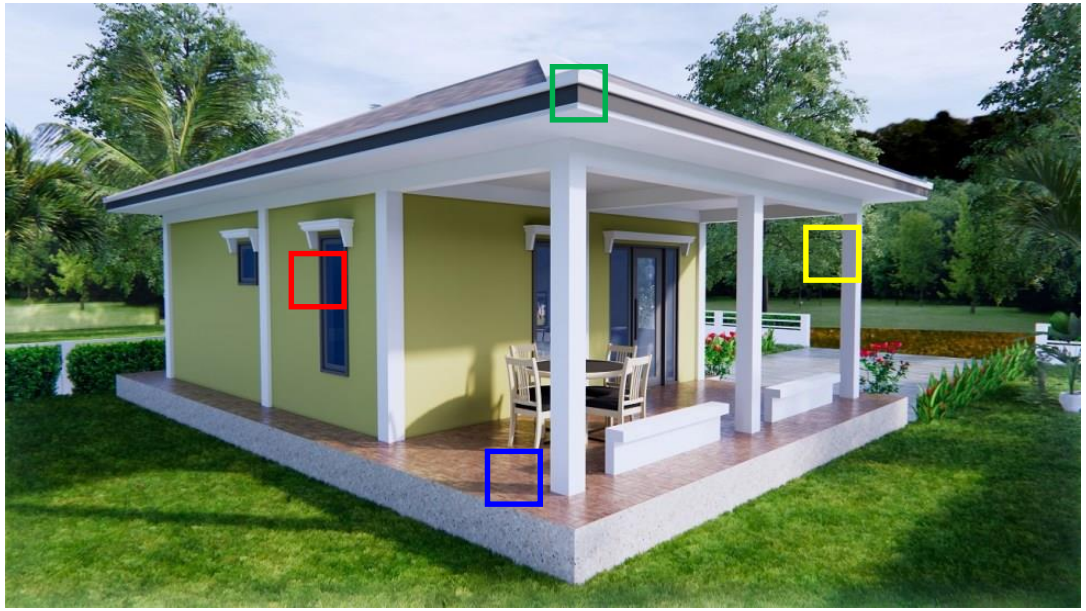
Origins of Edges

— Surface color discontinuity

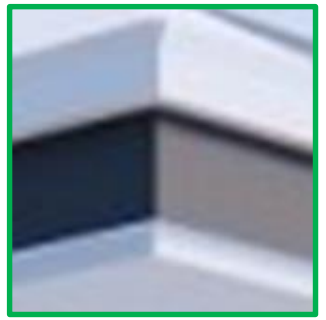
— Surface normal discontinuity

— Depth discontinuity

— Illumination discontinuity



Surface color discontinuity



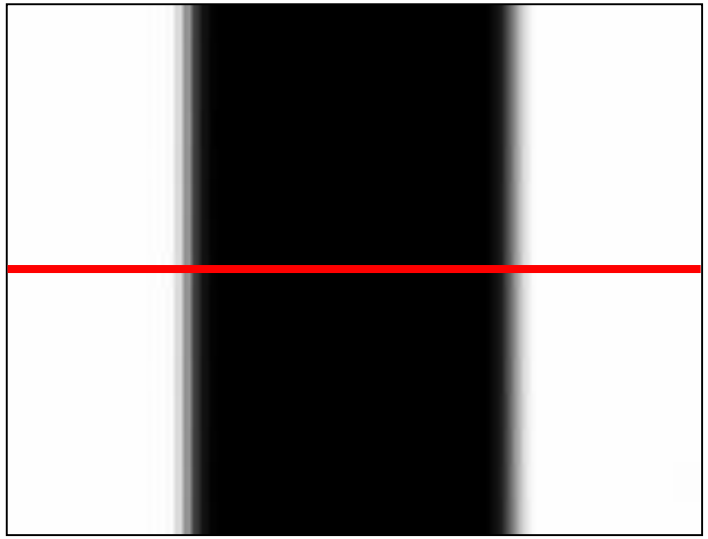
Surface normal discontinuity

Depth discontinuity

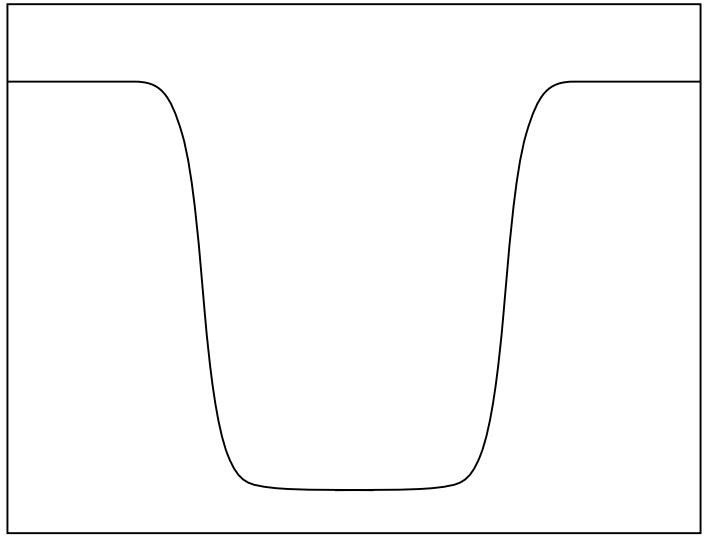
Illumination discontinuity

Characteristics of Edges

- An edge is a place of rapid change in the image intensity function

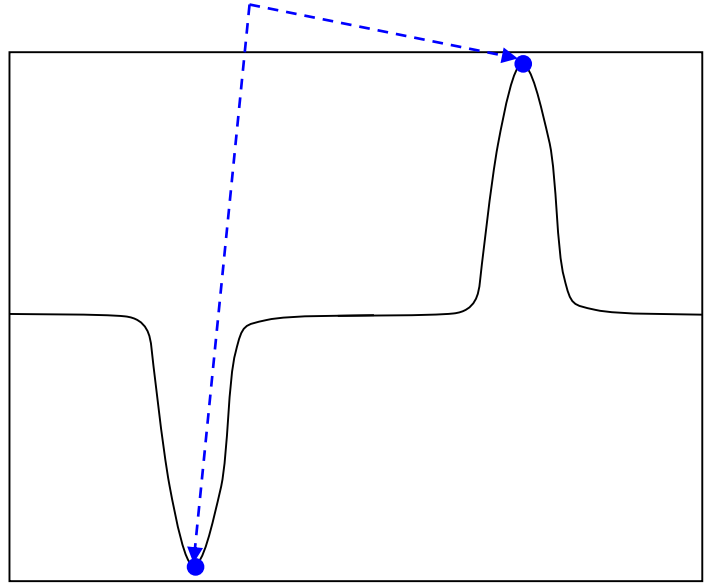


Image



Intensity function
along horizontal scanline

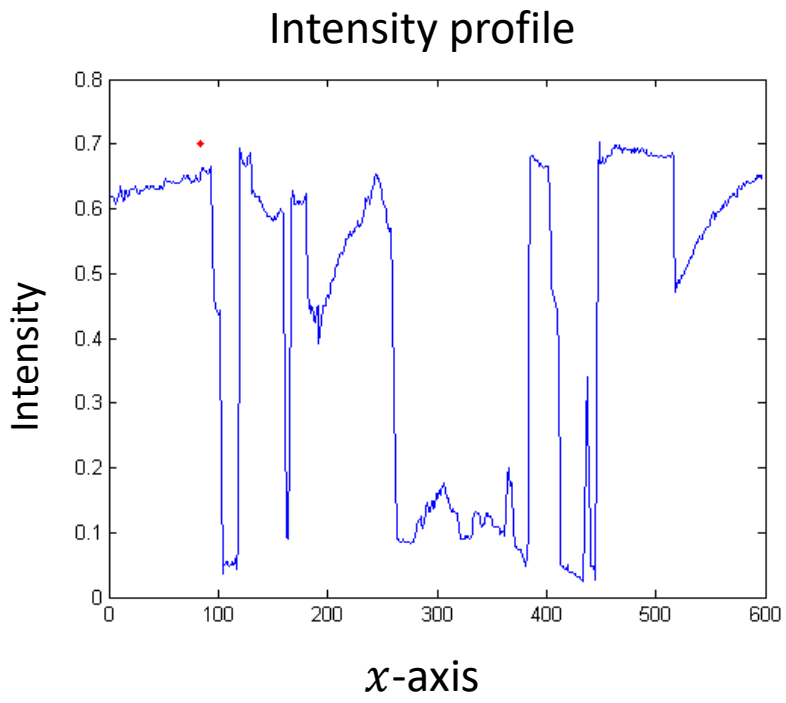
Edges corresponding to extreme of derivatives



1st derivative
in horizontal axis

Characteristics of Edges

- Intensity & gradient profiles



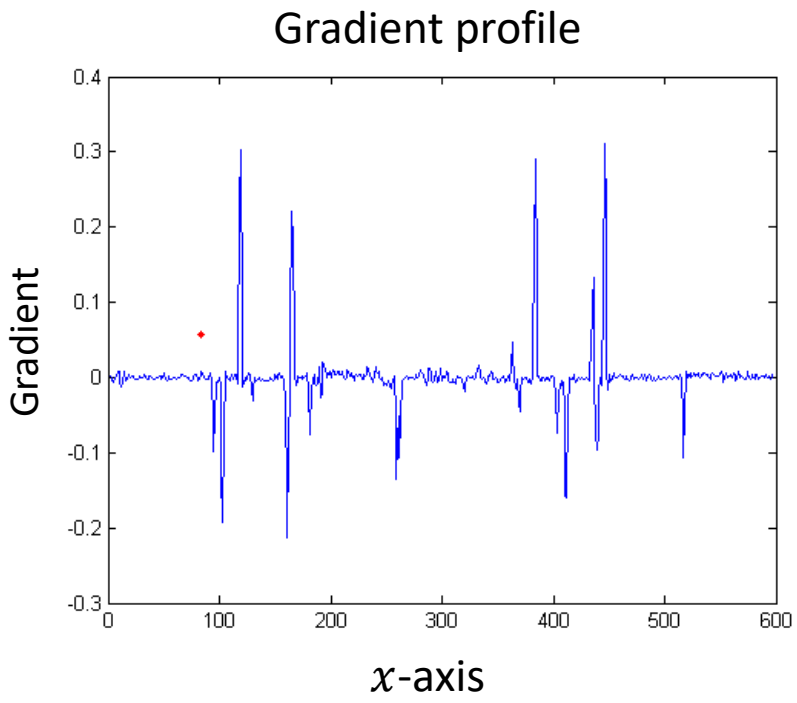
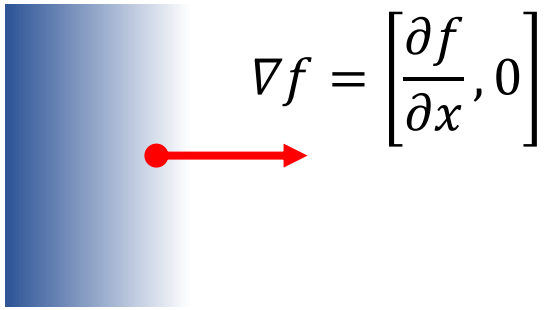
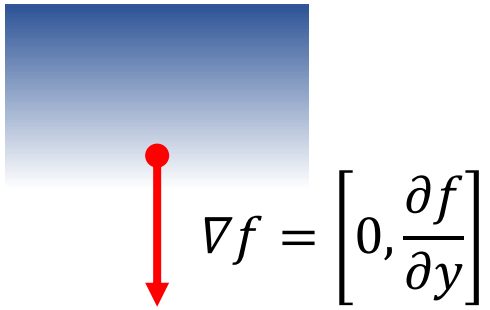


Image **Gradient**

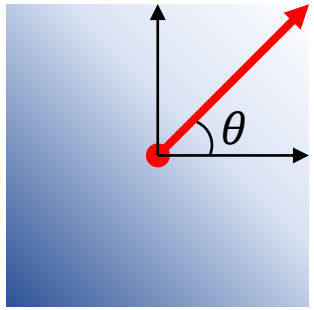
- **The Gradient of an Image:**



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

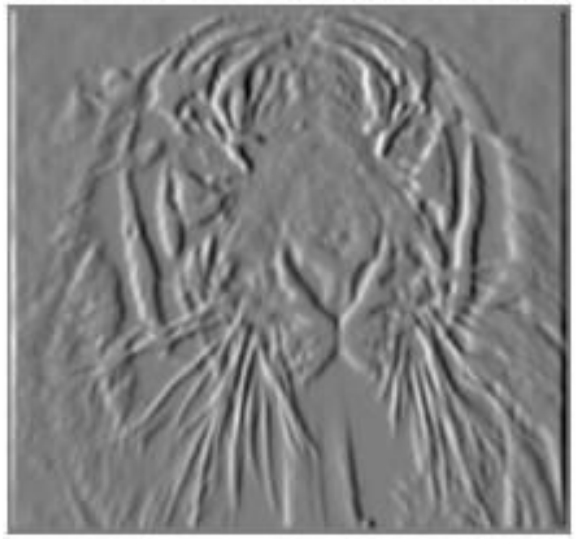
- **The gradient direction:** $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$
- **The gradient magnitude:** $\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$

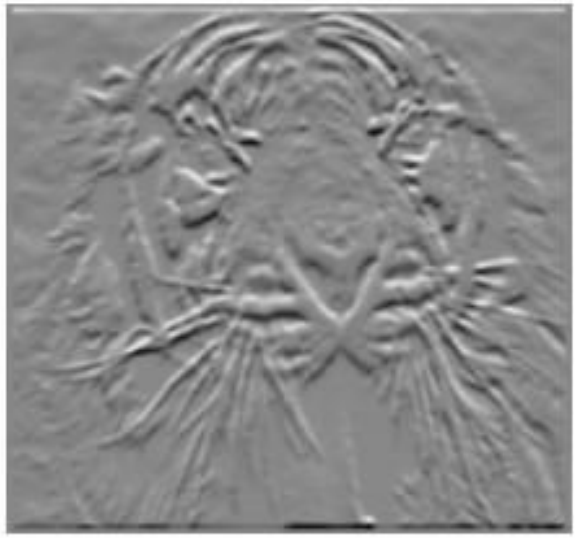
Gradient **Magnitude & Direction**

Original image

Gradient magnitude

The gradient
in the x-direction





The gradient
in the y-direction

Smoothing and Differentiation

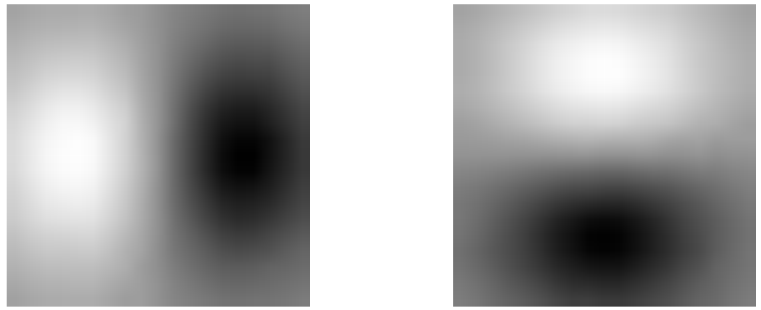
- **Edge:** a location with high gradient or derivative

- Need to something to reduce noise (high frequency) to taking derivative
- For this purpose, the **derivative of Gaussian (DoG)** filters can be used

— Because differentiation is convolution

— Convolution is associative

$$D \otimes \left(G \otimes I(x,y)\right) = (D \otimes G) \otimes I(x,y)$$

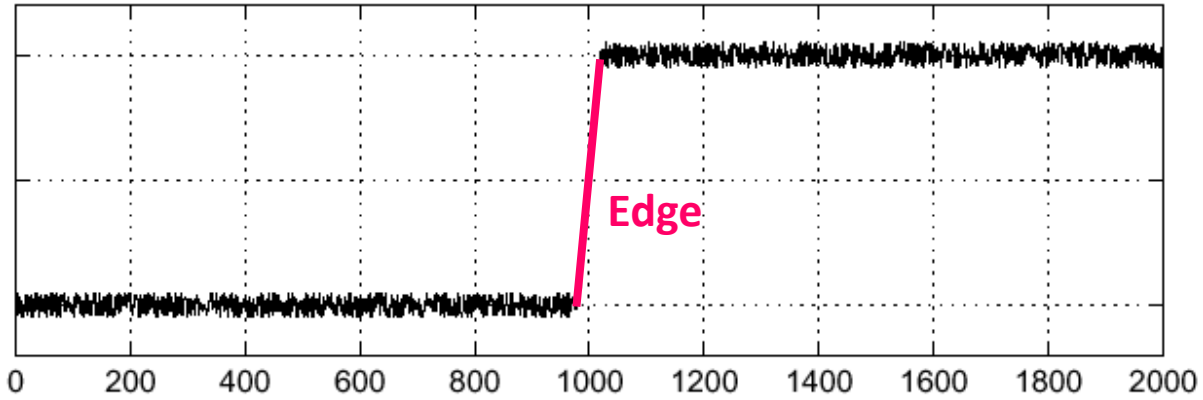


DoG filters in x and y direction

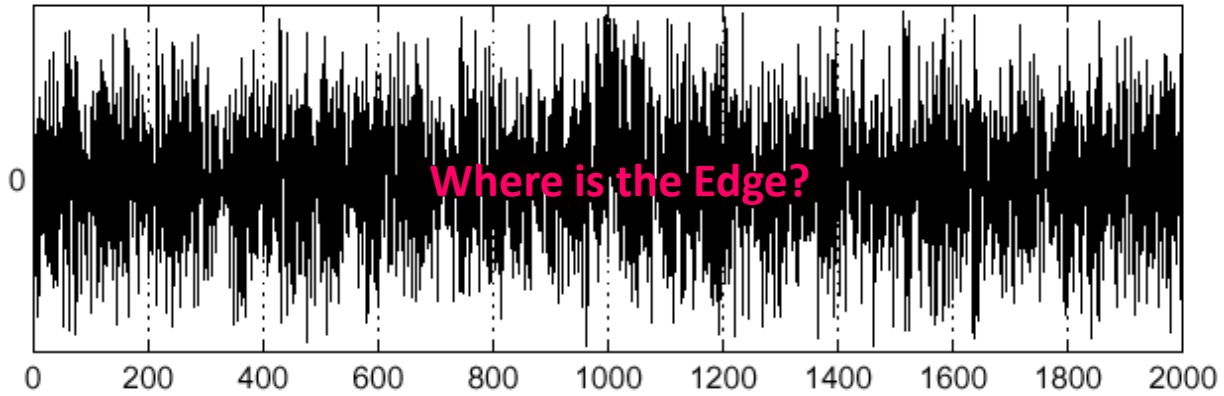
Example: **Derivative**

- Consider a row of pixels in an image:

$I(x, 200)$



$\frac{d}{dx} I(x, 200)$



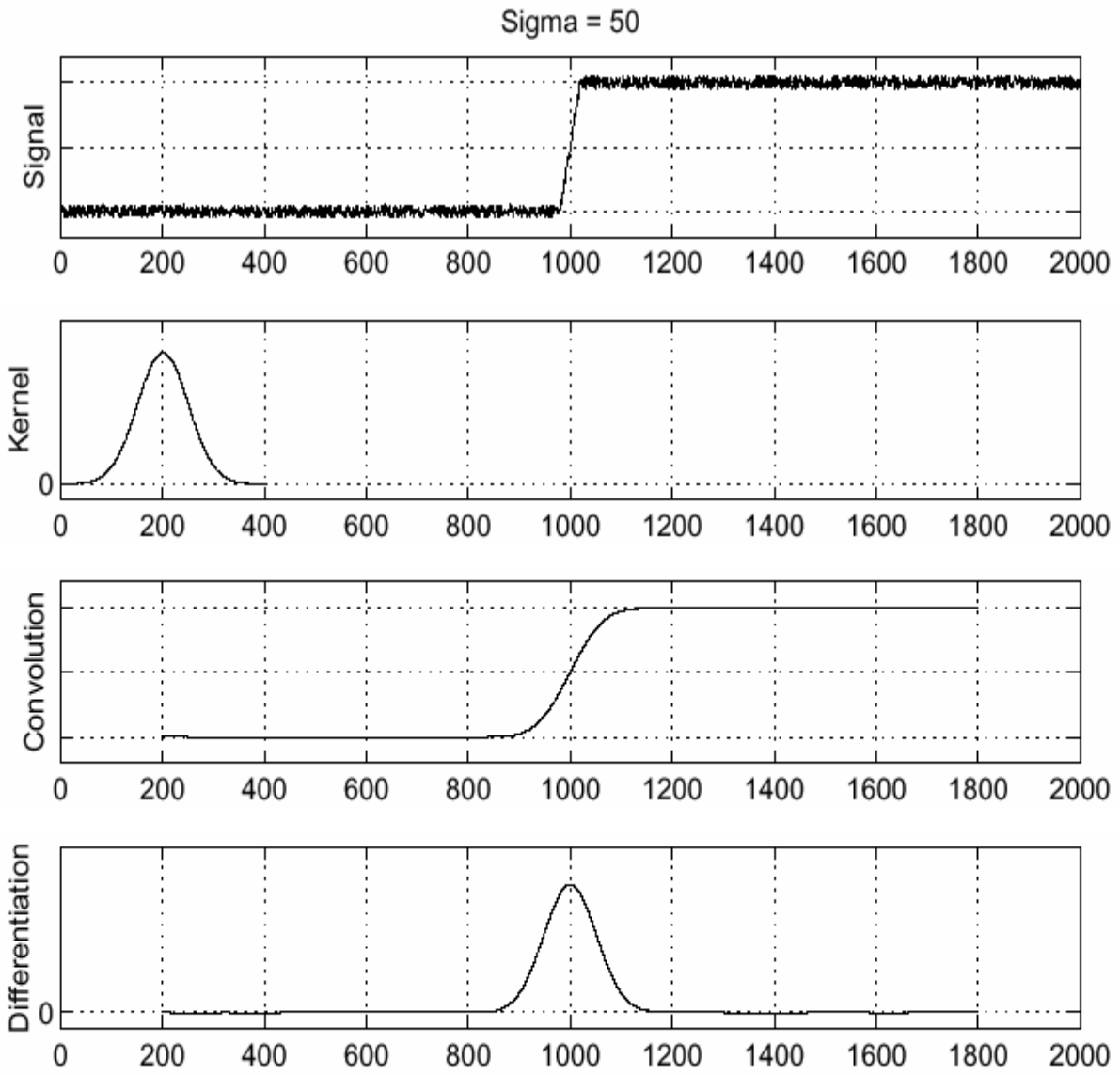
Example: **Smoothing + Derivative**

$I(x, y)$

G

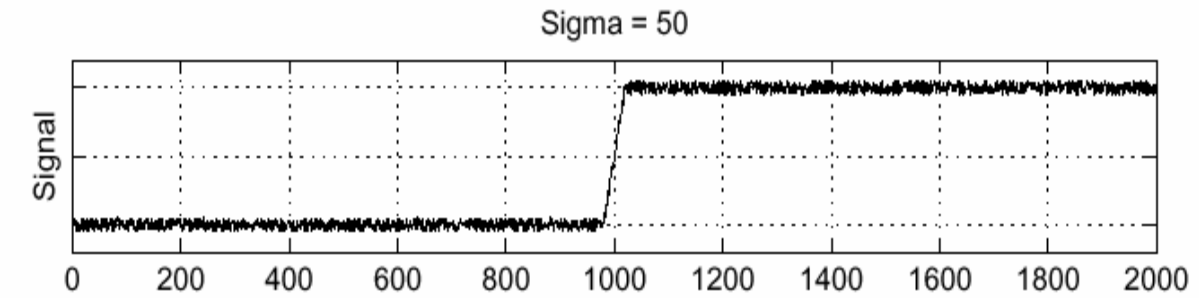
$G \otimes I(x, y)$

$\frac{\partial}{\partial x}\left(G \otimes I(x, y)\right)$

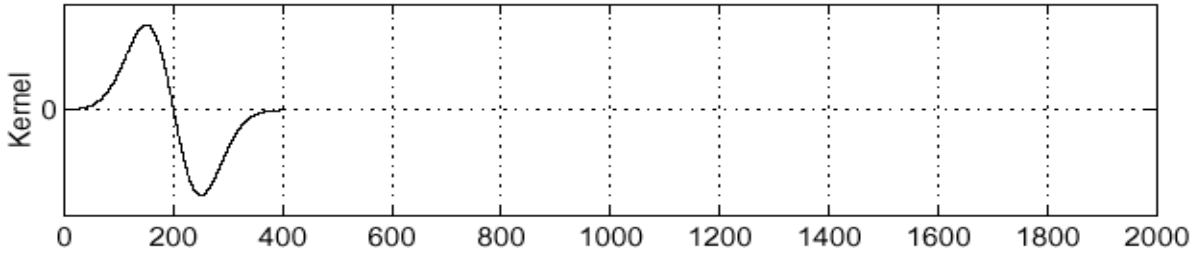


Example: **Derivatives of Gaussian (DoG)**

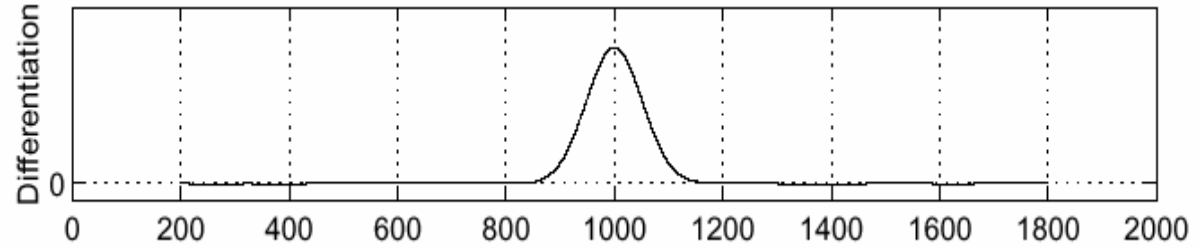
$I(x, y)$



$\frac{\partial}{\partial x} G$



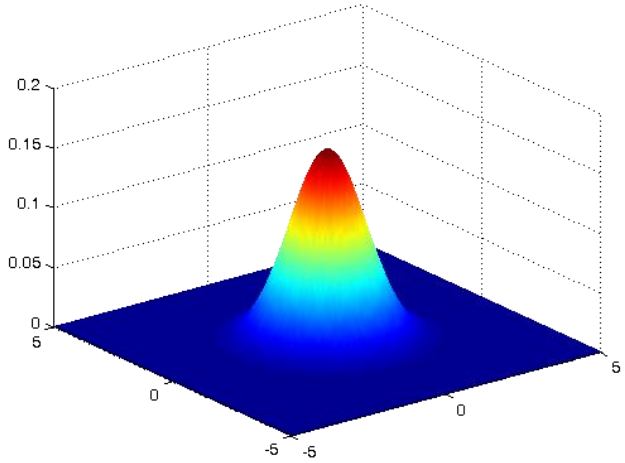
$\frac{\partial}{\partial x} G \otimes I(x, y)$



Partial **Derivatives of Gaussian**

- **Base:** Gaussian function

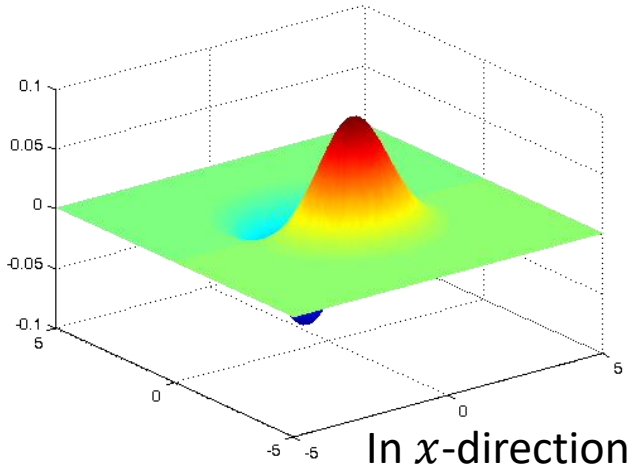
$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

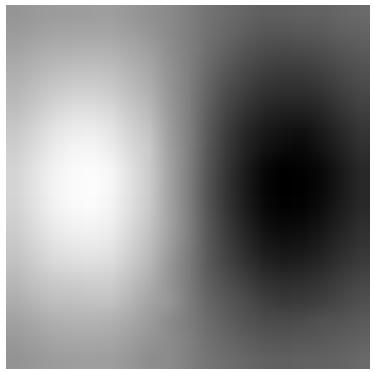


Gaussian function (Base)

- **1st order derivative of Gaussian**

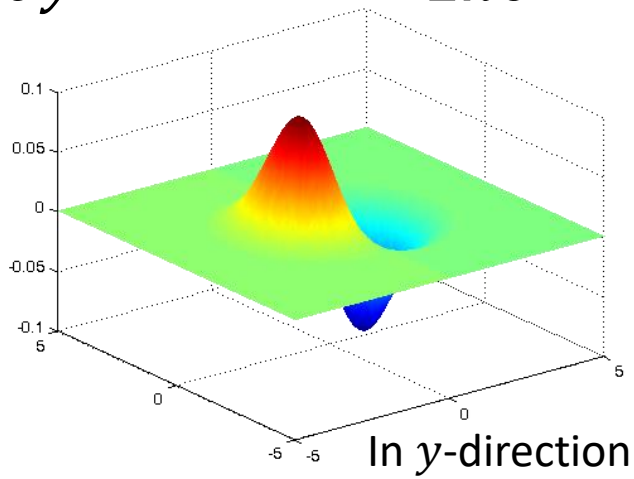
$$\frac{\partial}{\partial x} G(x, y, \sigma) = \frac{x}{2\pi\sigma^4} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

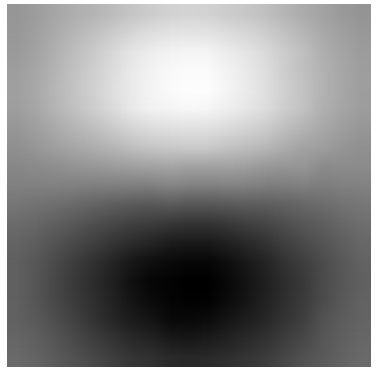




In x-direction

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





In y-direction

Sobel Edge Detector

① Use **central differencing** to compute gradient image

② **Threshold** to obtain edges

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{I} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \left(\begin{bmatrix} -1 & 0 & +1 \end{bmatrix} * \mathbf{I} \right)$$

Sobel operator for horizontal changes

Smoothing

Derivative

$$\mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{I} = \begin{bmatrix} +1 \\ 0 \\ -1 \end{bmatrix} * \left(\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \mathbf{I} \right)$$

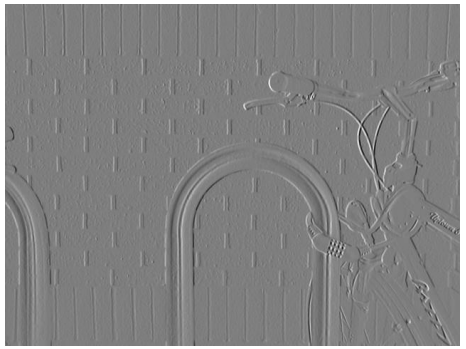
Sobel operator for vertical changes

Derivative

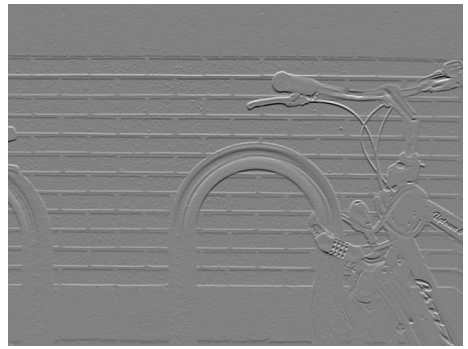
Smoothing

Original image

Gradient magnitude



x-gradient



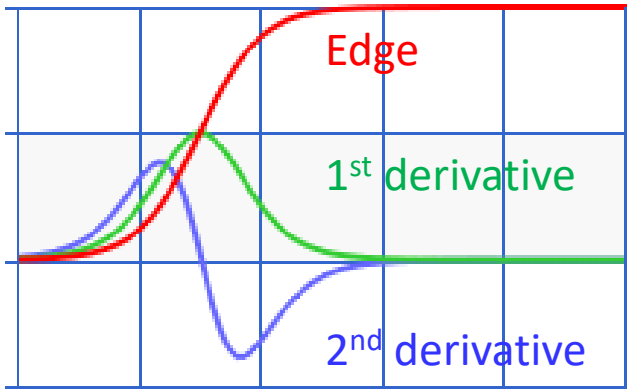
y-gradient

Laplacian of Gaussian (LoG)

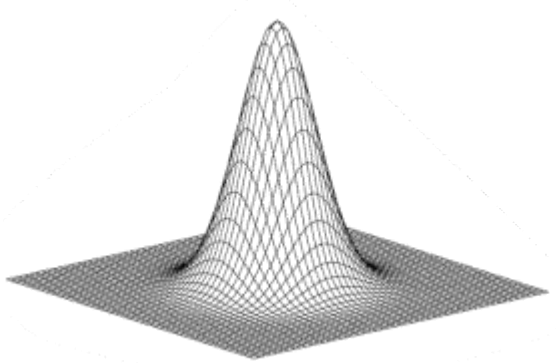
- **Two Generic Edge Detection Approaches:**

— Local extrema of a first derivative operator

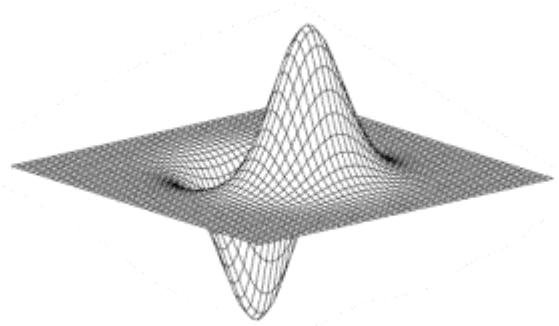
— Zero crossings of a second derivative operator



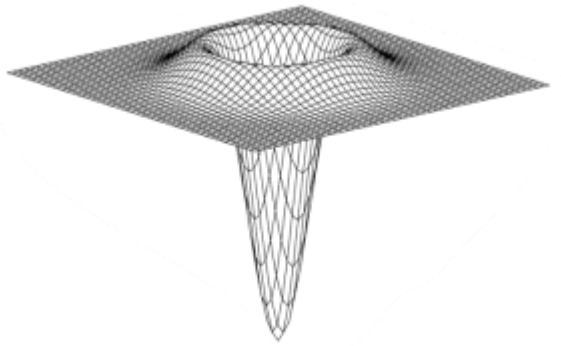
- **Laplacian of Gaussian** is based on **a zero crossings of a second derivative** operator method



G_σ



∇G_σ



$\nabla^2 G_\sigma$

Laplacian of Gaussian (LoG)

① **Gaussian** for smoothing

② **Laplacian** (∇^2) for differentiation

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

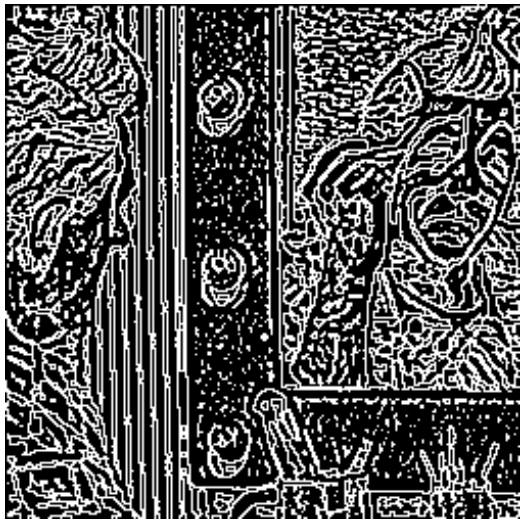
③ **Locate zero crossings** in the Laplacian of the Gaussian $(\nabla^2 G)$

$$\nabla^2 G(x,y;\sigma) = \left[\frac{x^2+y^2}{\sigma^4} - \frac{2}{\sigma^2}\right] G(x,y;\sigma)$$

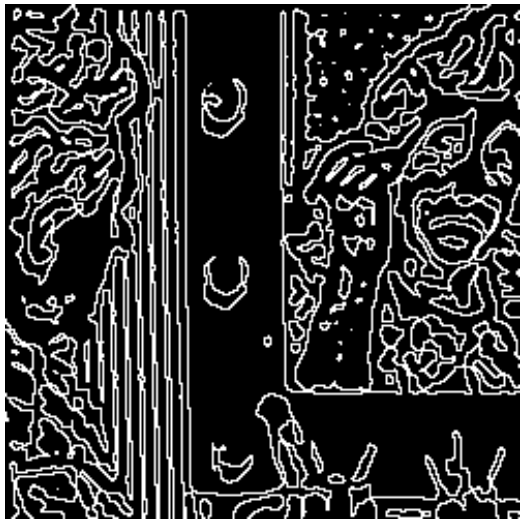
Laplacian of Gaussian (LoG)

Original image

Results of LoG



$\sigma = 1$



$\sigma = 2$

Zero crossings

Canny Edge Detector

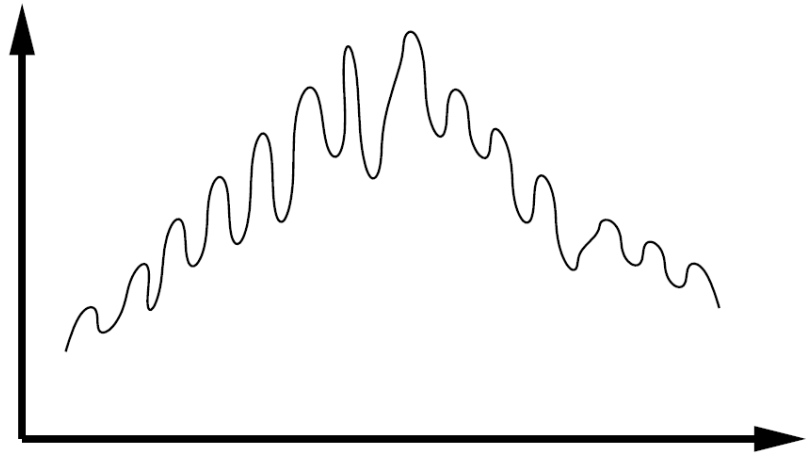
- **Canny edge detector** is based on a **local extrema of a first derivative** operator approach

- **Design Criteria:**

— Good detection
  - Low error rate for omissions (missed edges)
  - Low error rate for commissions (false positive)

— Good localization

— Single response to a given edge
  - Eliminate multiple responses to a single edge



Q: How many edges are?

Q: What is the position of each edge?

Canny Edge Detector

① Apply **directional derivatives of Gaussian**

② Compute **gradient magnitude** and **gradient direction**

③ **Non-maximum suppression (NMS)**

— Thin multi-pixel wide “ridges” down to single pixel width

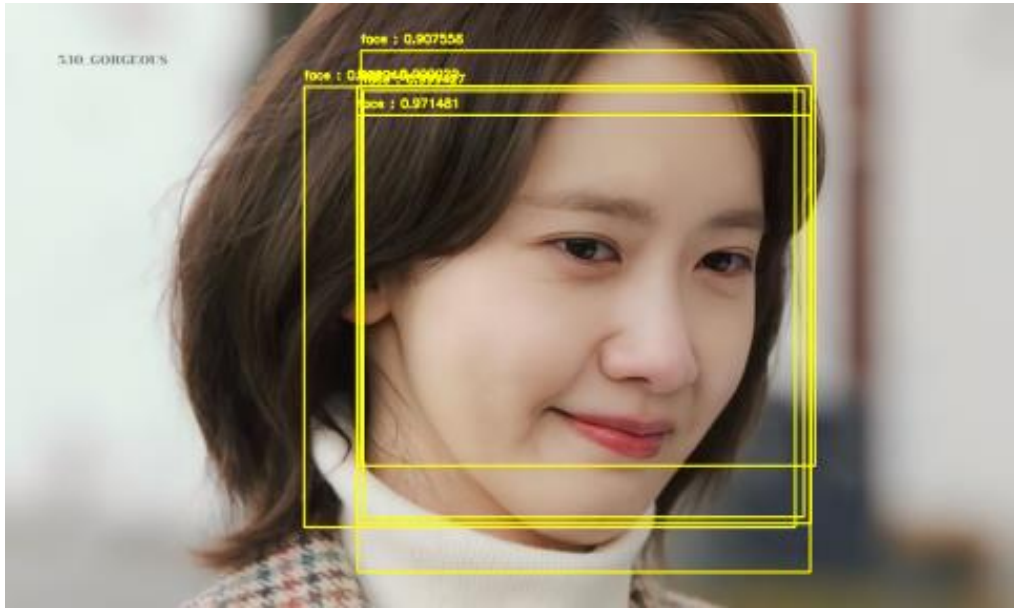
④ **Link** and **threshold**

— Low and high edge-strength thresholds

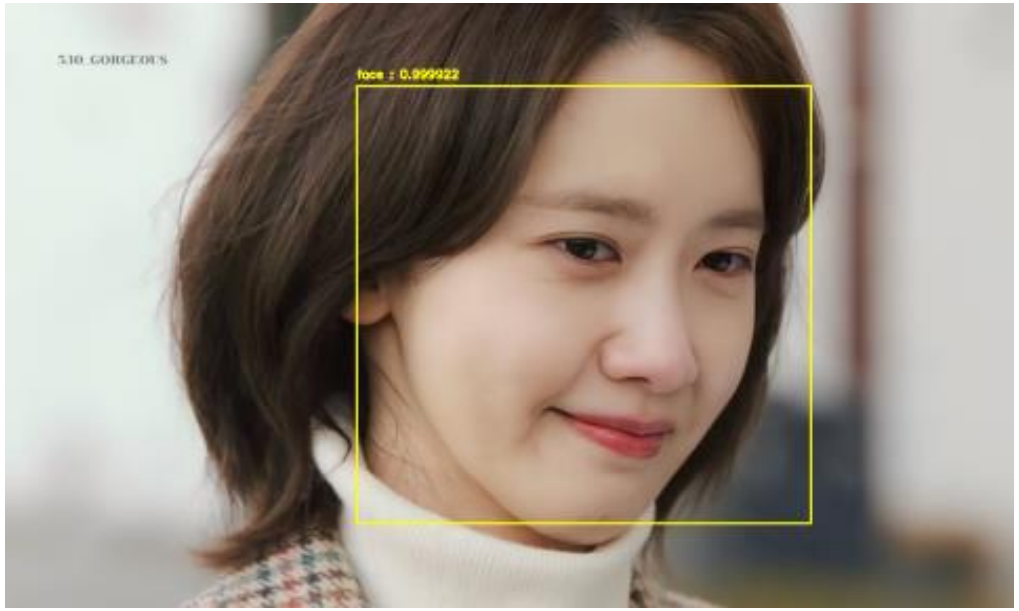
— Accept all edges over low threshold that are connected to edge over high threshold

Canny Edge Detector: **Non-Maximum Suppression**

- Suppress near-by similar detections to obtain one “true” result



Before non-maximum suppression



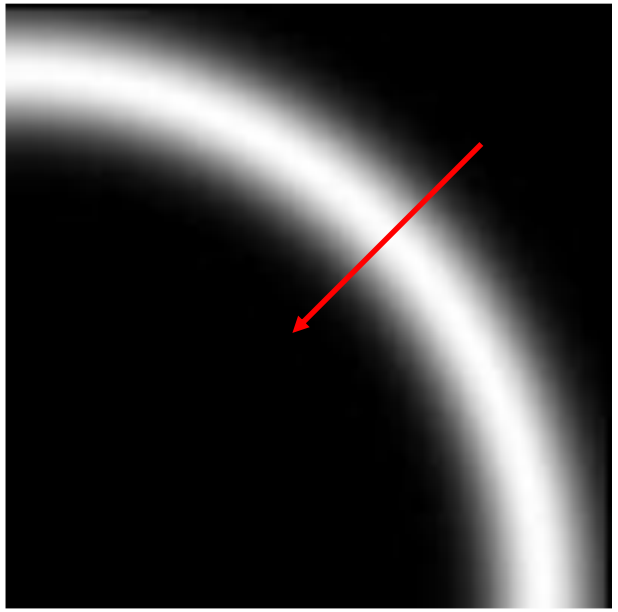
After non-maximum suppression

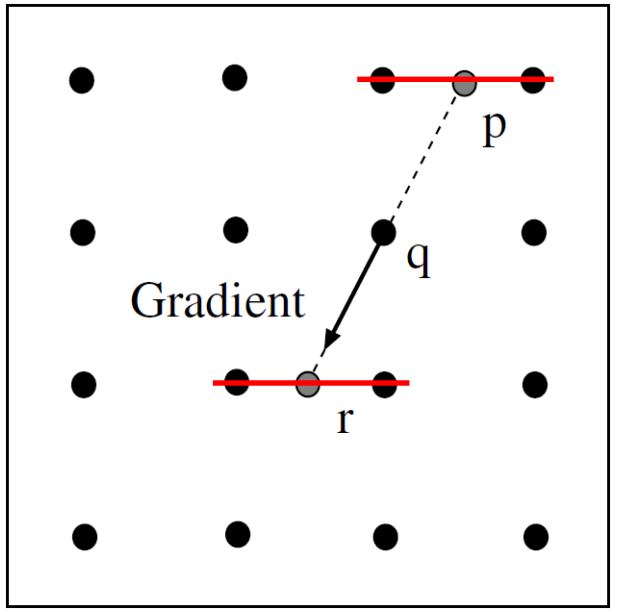
In face detection application

Canny Edge Detector: **Non-Maximum Suppression**

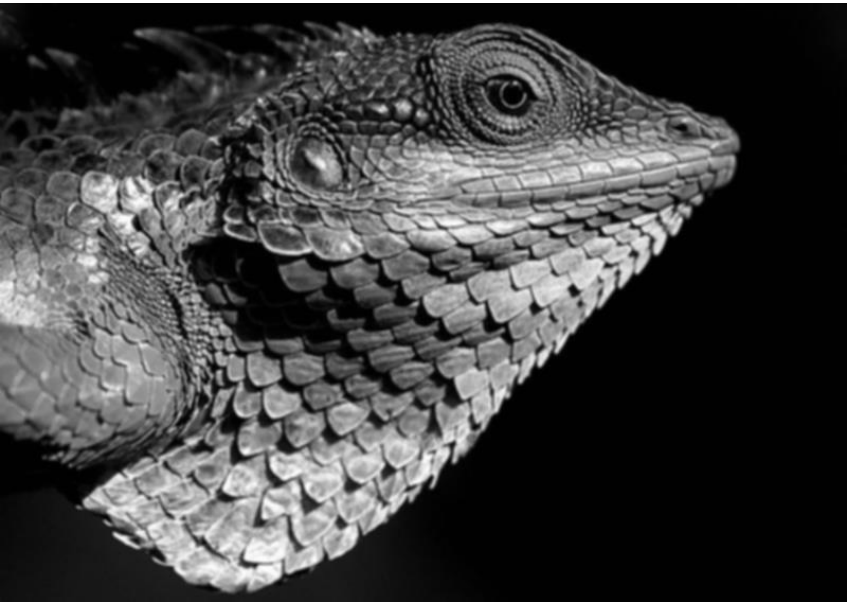
- Suppress near-by similar detections to obtain one “true” result
- Select the image **maximum point** across the width of the edge

- If the value at q is larger than p and r, q is an edge point

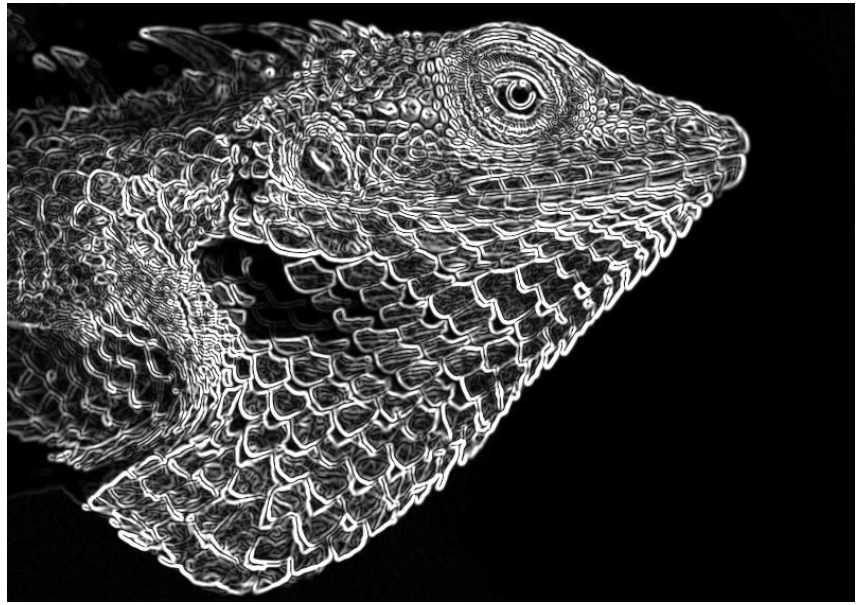




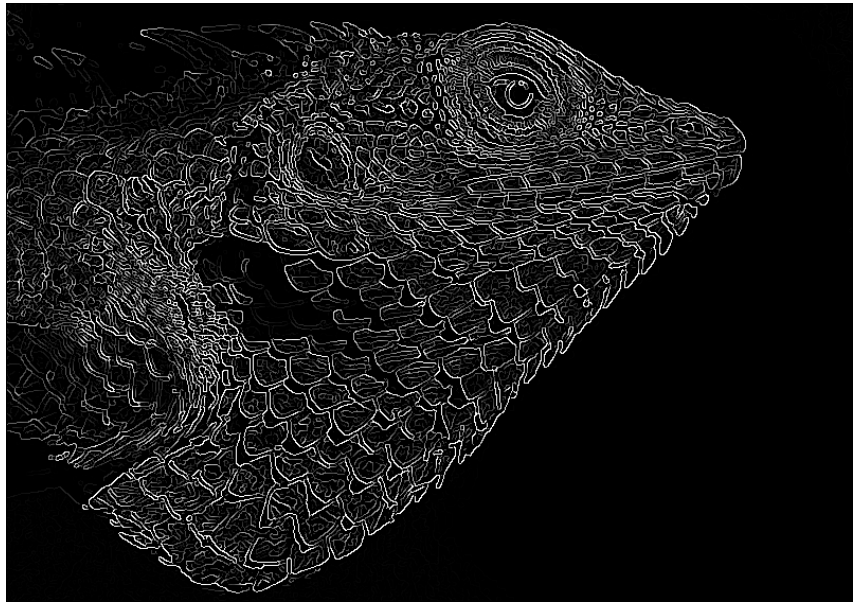
Canny Edge Detector: **Non-Maximum Suppression**



Original image



Gradient magnitude



Result of non-maximum suppression

https://en.wikipedia.org/wiki/Canny_edge_detector

Canny Edge Detector: **Linking** Edges

- A way to deal with broken edge chains is to use edge hysteresis

- Maintain two thresholds Th_{high} and Th_{low}

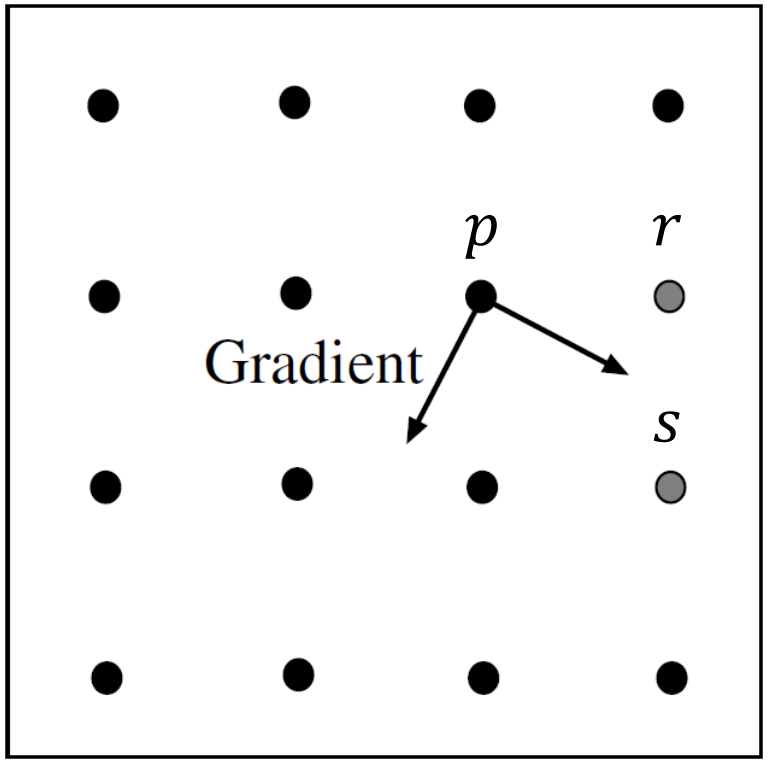
— Use Th_{high} to find strong edges to start edge chain

— Use Th_{low} to find weak edges which continue edge chain

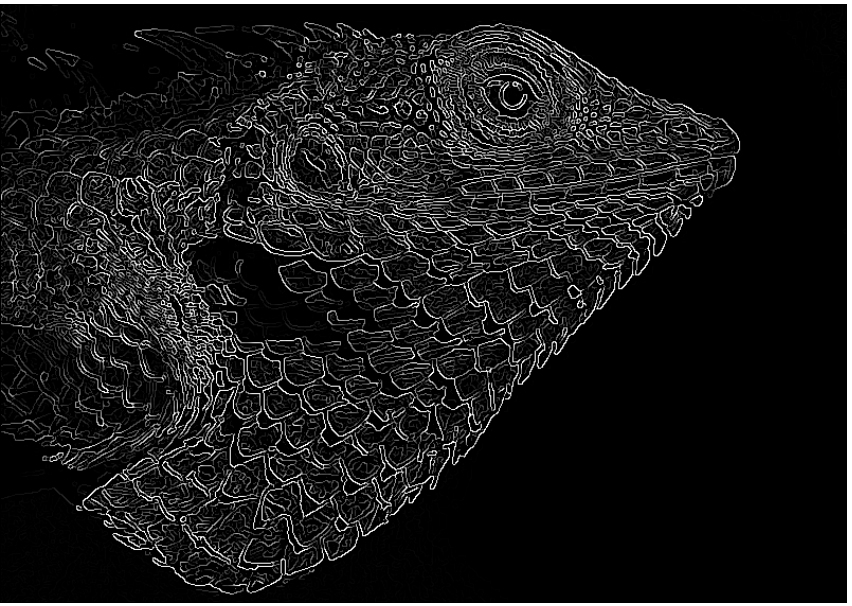
- Typical ratio of thresholds is ${Th_{high}}/{Th_{low}} = 2$

Canny Edge Detector: **Linking** Edges

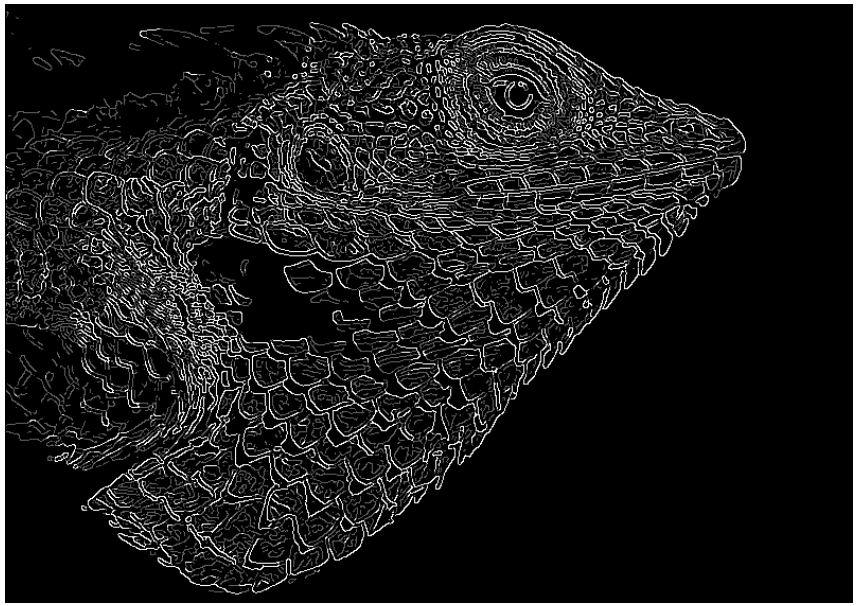
- Assume the marked point p is an edge point
- Take the normal to the gradient at that point and use this to predict continuation points, either r or s



Canny Edge Detector: **Edge Hysteresis**

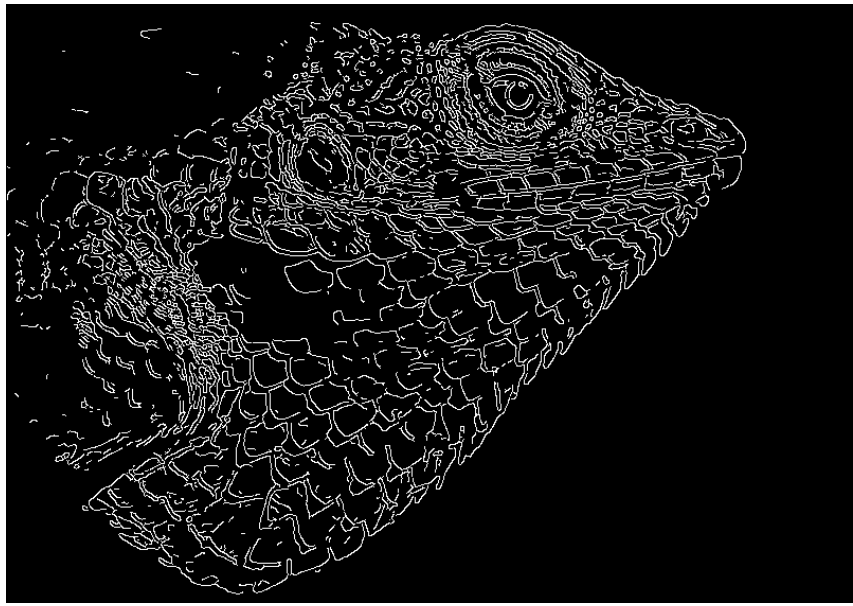


Result of non-maximum suppression



Result of double thresholding

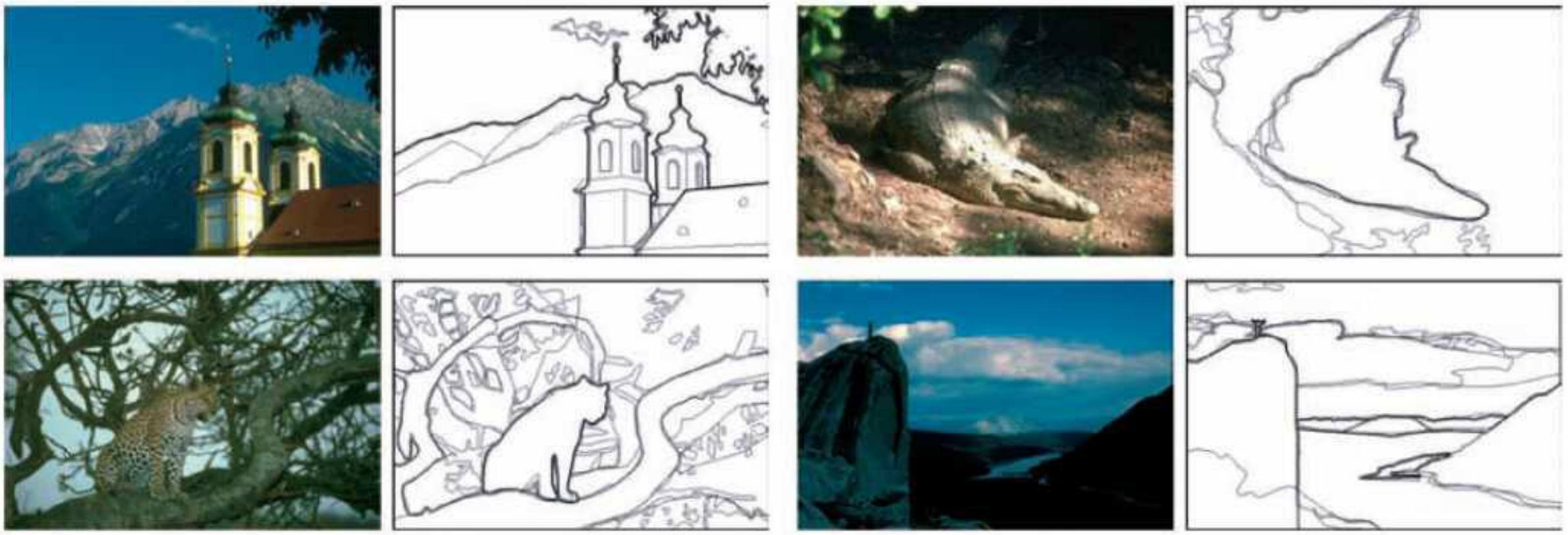
- Weak pixels : 0.1 < gradient < 0.3
- Strong pixels: 0.3 < gradient



Result of edge hysteresis

Boundaries **in Human Vision**

- Each image shows multiple human-marked boundaries
- Pixels are darker where more humans marked a boundary



Martin et al., Learning to Detect Natural Image Boundaries using Local Brightness, Color, and Texture Cues, **IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI), 2004**

Summary: **Features – Edge Detection**

- **Physical properties** of a 3D scene cause **edges** in an image:

— Surface color discontinuity

— Surface normal discontinuity

— Depth discontinuity

— Illumination discontinuity

- **Two generic edge detection** approaches:

— Local extrema of a first derivative operator → **Canny edge detector**

— Zero crossings of a second derivative operator → **LoG detector**

Characteristics of **Good Features**

— **Locality**
- Features are local, robust to occlusion and clutter

— **Accurate**
- Precise localization

— **Robustness**
- Noise, blur, compression, etc. do not have a big impact on the feature

— **Distinctiveness**
- Individual features can be easily matched

— **Efficiency**
- Close to real-time performance

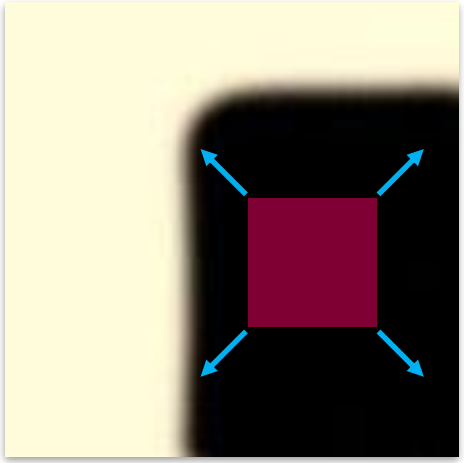
Corner

- **Key Property:**

— In the region around a corner, image gradient has two or more dominant directions

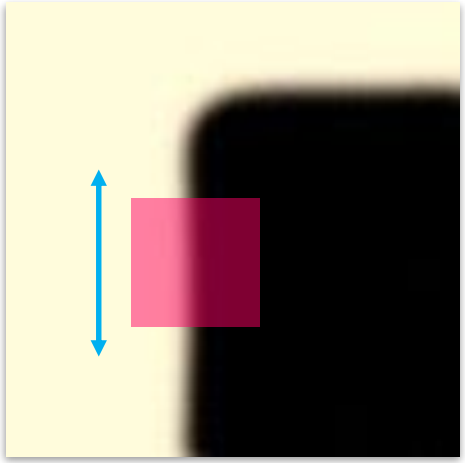
— Corners are robust and distinctive

Flat



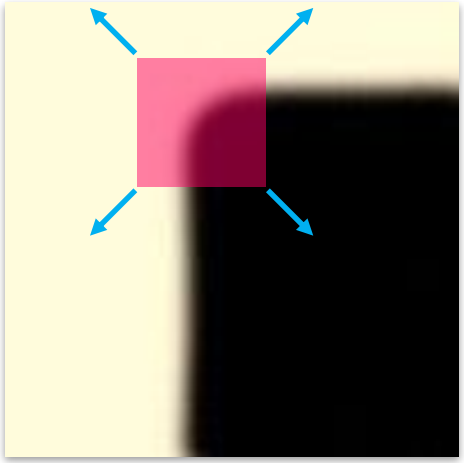
No changes
in all directions

Edge



No changes
along the edge direction

Corner



Significant changes
in all directions

C. Harris and M. Stephens, A combined Corner and Edge Detector, **Alvey Vision Conference**, 1988

Corner Detection

- Edge detectors perform poorly at corners

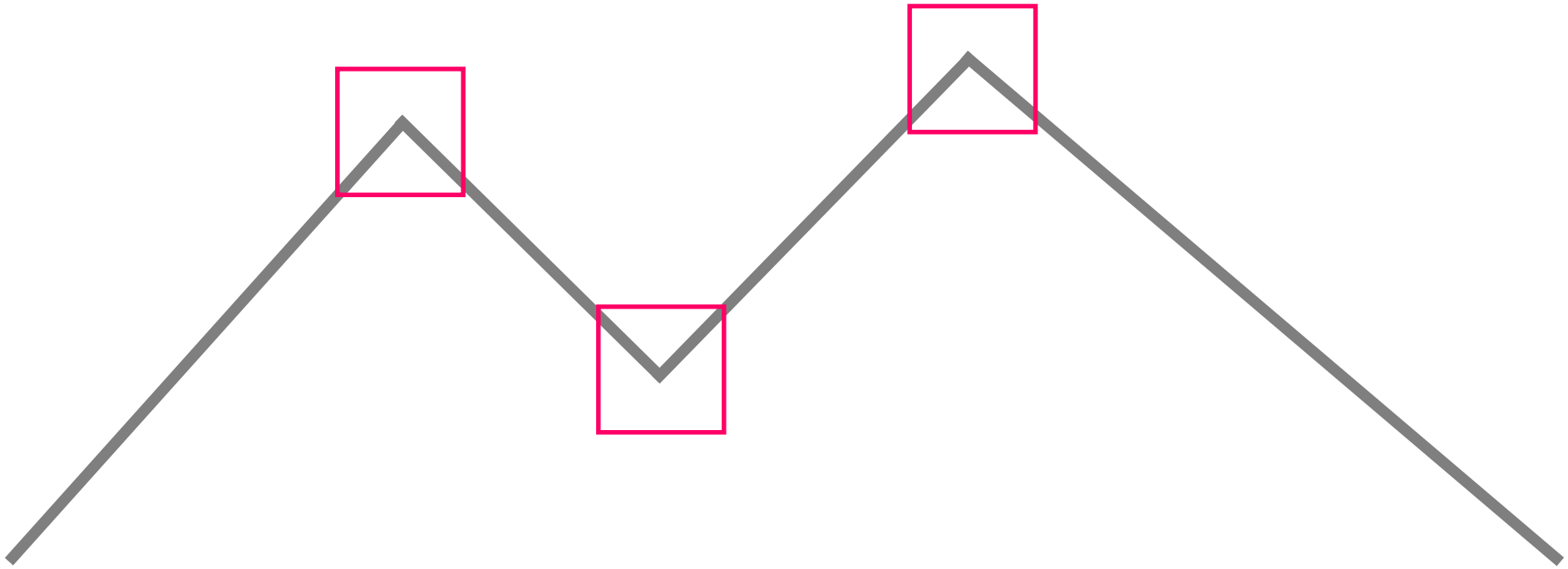
- **Observations:**

— The gradient is ill-defined exactly at a corner

— Near a corner, the gradient has two or more distinct values

How to Find a Corner

- Easily recognized by looking through a small window
- Shifting the window should give large change in intensity



Harris Corner Detection

① Compute **image gradient** over small region

② Compute the **covariance matrix**

③ Compute **eigenvectors** and **eigenvalues**

④ Use **threshold on eigenvalues** to detect corners

Harris Corner Detection: ① **Image Gradient**

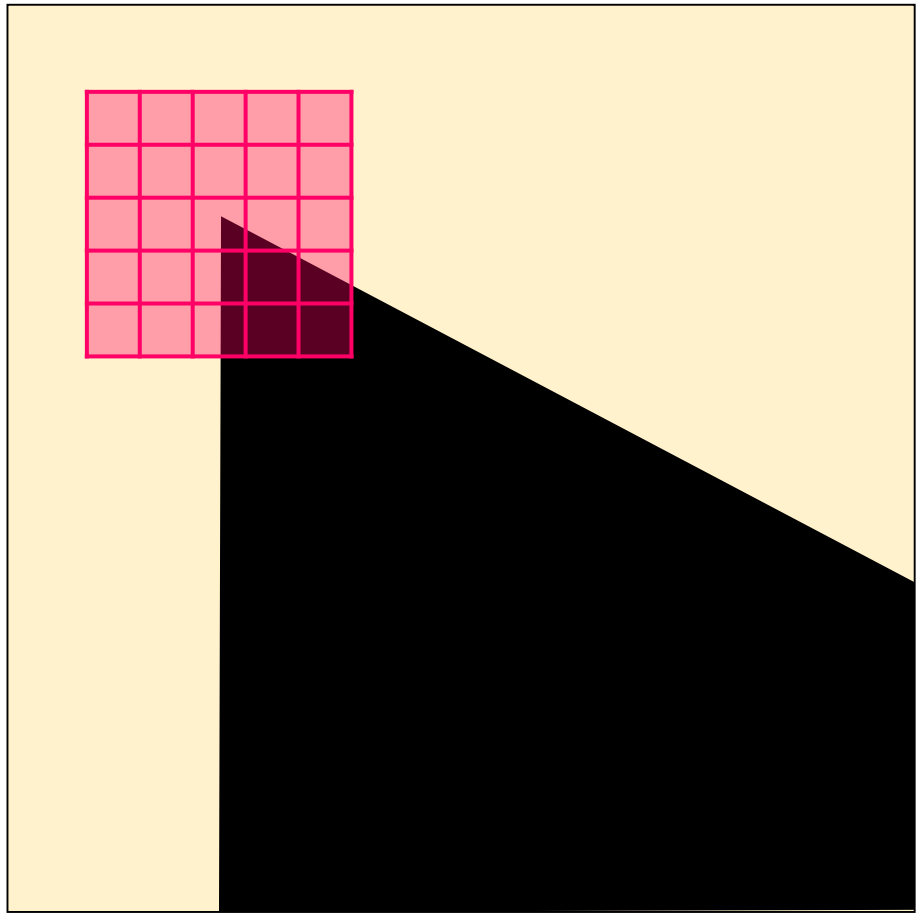
- Compute **image gradient** over small region

— Gradient in x direction:

$$I_y = \frac{\partial I}{\partial y}$$

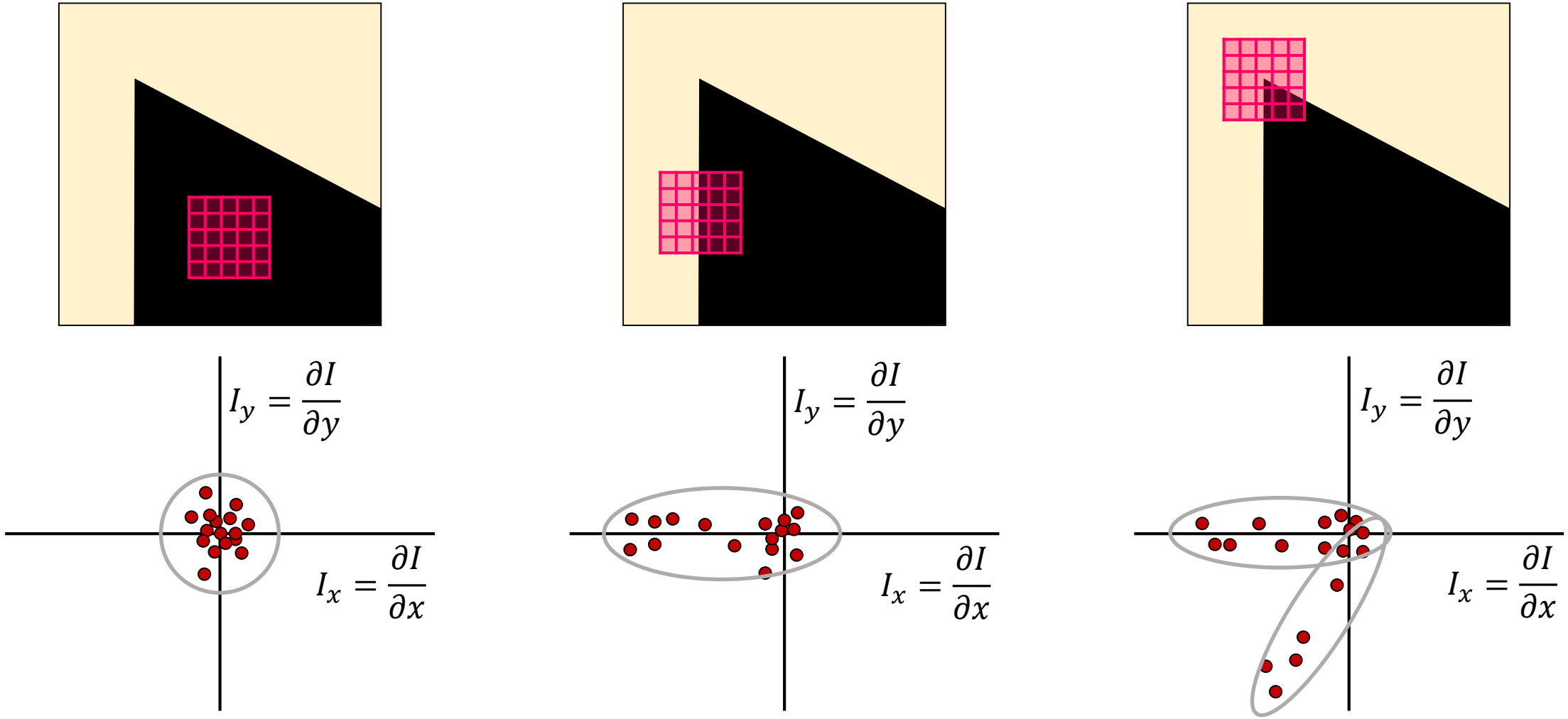
— Gradient in y direction:

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



Harris Corner Detection: ① **Image Gradient**

- Distribution reveals **edge orientation** and **magnitude**



Harris Corner Detection: ② **Covariance Matrix**

- Compute the **covariance matrix**

$$\mathbf{M} = \begin{bmatrix} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\ \sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{bmatrix}$$

Harris Corner Detection: ② **Covariance Matrix**

- Compute the **covariance matrix**

Sum over local window region around corner

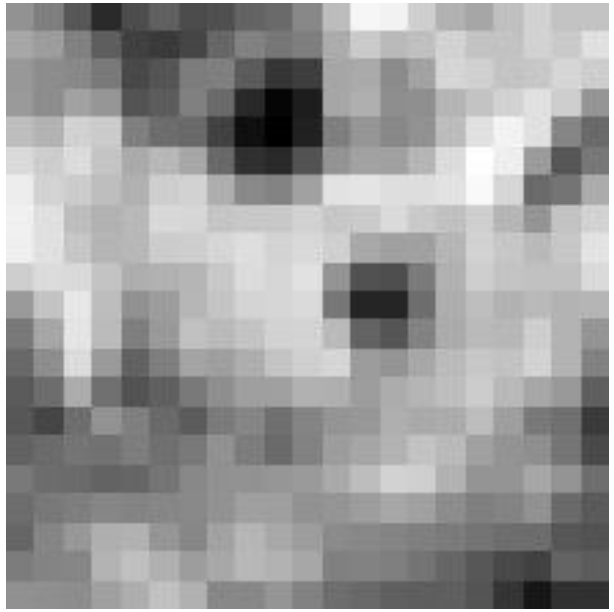
Gradient with respect to each direction (x or y)

$$\mathbf{M} = \begin{bmatrix} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\ \sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{bmatrix}$$

Harris Corner Detection: ② **Covariance Matrix**

- Change of intensity for the shift value $[u, v]$, error function:

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

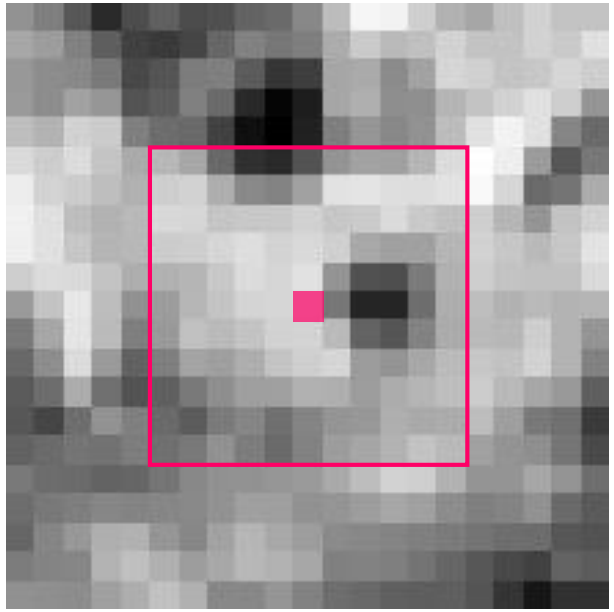


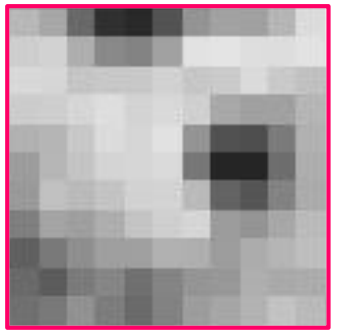
Harris Corner Detection: ② **Covariance Matrix**

- Change of intensity for the shift value $[u, v]$, error function:

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

Center pixel





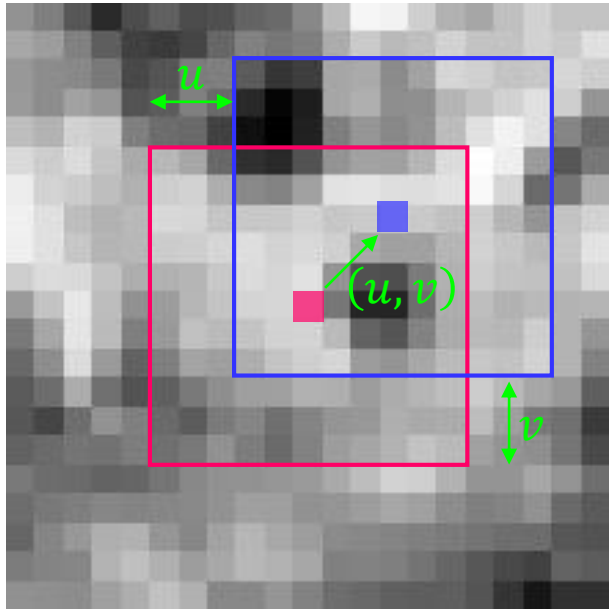
$I(x,y)$

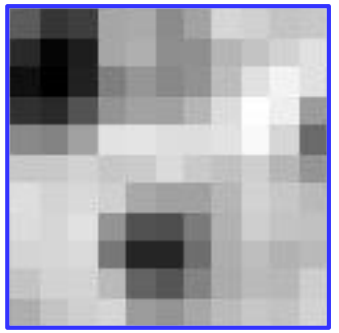
Harris Corner Detection: ② **Covariance Matrix**

- Change of intensity for the shift value $[u, v]$, error function:

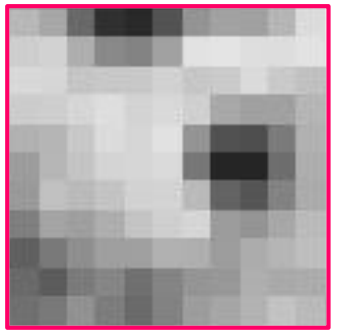
$$E(u,v) = \sum_{(x,y)\in P} w(x,y)[I(x+u, y+v) - I(x,y)]^2$$

Neighboring pixels — Center pixel





$I(x+u, y+v)$



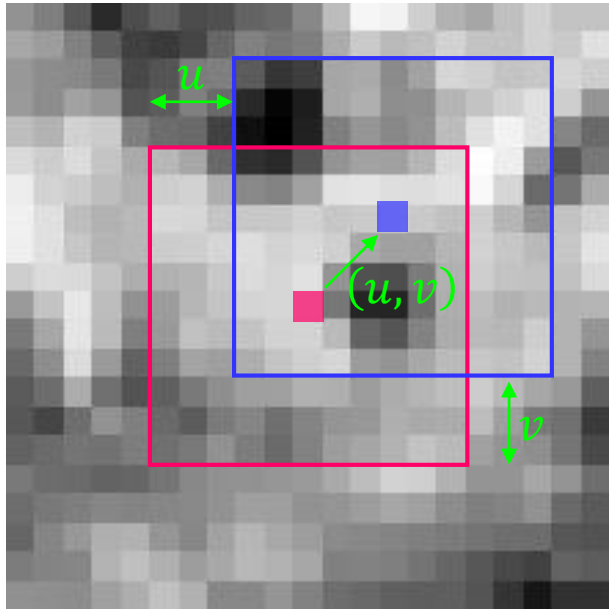
$I(x,y)$

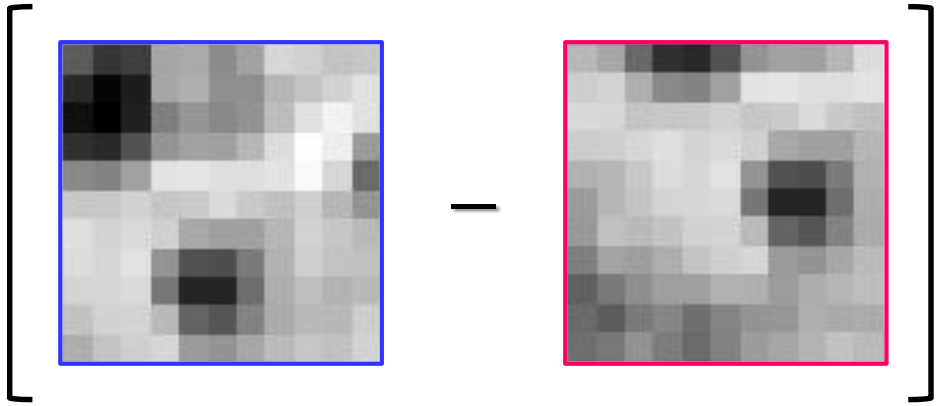
Harris Corner Detection: ② **Covariance Matrix**

- Change of intensity for the shift value $[u, v]$, error function:

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

Neighboring pixels — Center pixel





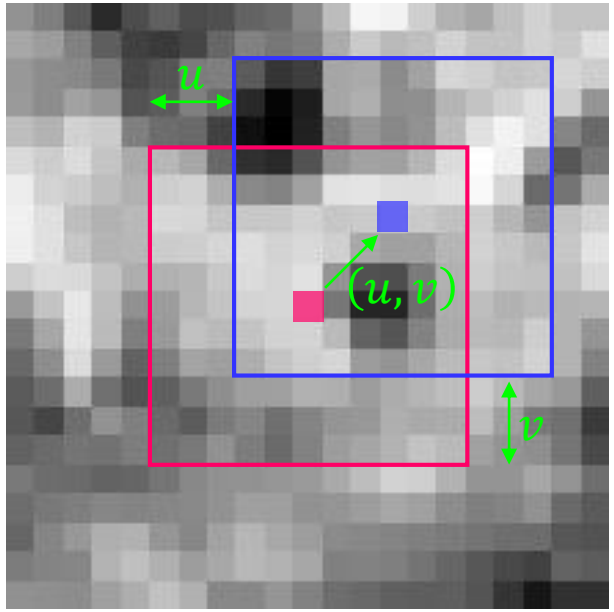
$I(x+u, y+v)$ $I(x,y)$

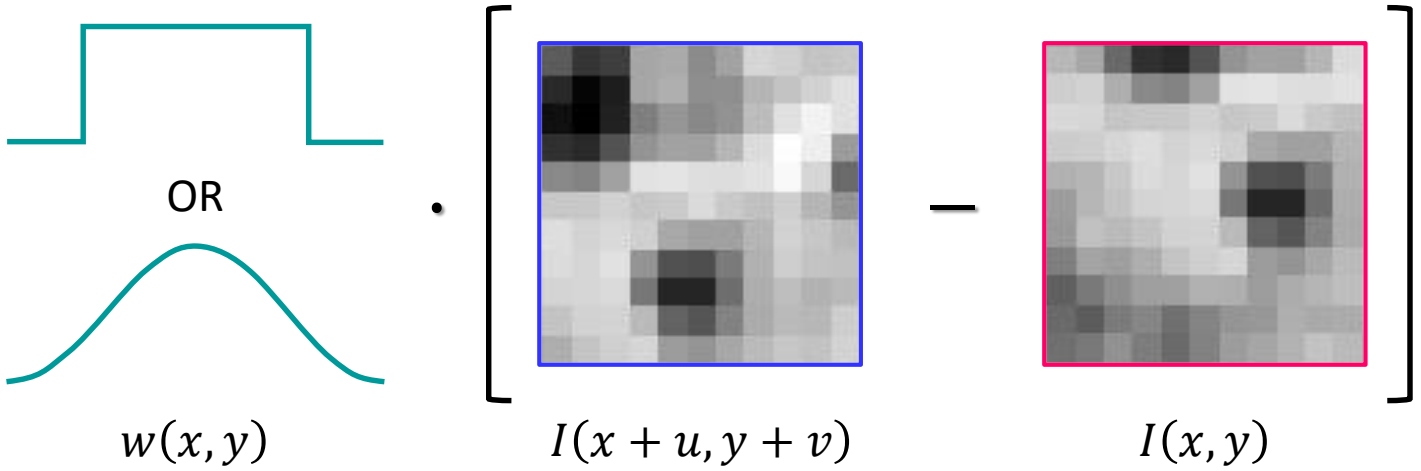
Harris Corner Detection: ② **Covariance Matrix**

- Change of intensity for the shift value $[u, v]$, error function:

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

Window function — Neighboring pixels — Center pixel





Harris Corner Detection: ② **Covariance Matrix**

- Change of intensity for the shift value $[u, v]$, error function:

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

$$\approx \sum_{(x,y)\in P} w(x,y)\left[\left(\frac{\partial I}{\partial x}u\right)^2 + \left(\frac{\partial I}{\partial y}v\right)^2 + 2\frac{\partial I}{\partial x}\frac{\partial I}{\partial y}uv\right]$$

Harris Corner Detection: ② **Covariance Matrix**

- Change of intensity for the shift value $[u, v]$, error function:

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

$$\approx \sum_{(x,y)\in P} w(x,y)\left[\left(\frac{\partial I}{\partial x}u\right)^2 + \left(\frac{\partial I}{\partial y}v\right)^2 + 2\frac{\partial I}{\partial x}\frac{\partial I}{\partial y}uv\right]$$

$$= \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} \sum_{p\in P}\left(\frac{\partial I}{\partial x}\right)^2 & \sum_{p\in P}\frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \sum_{p\in P}\frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \sum_{p\in P}\left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

where $w(x,y) = 1$

Harris Corner Detection: ② **Covariance Matrix**

- Change of intensity for the shift value $[u, v]$, error function:

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

$$\approx \sum_{(x,y)\in P} w(x,y)\left[\left(\frac{\partial I}{\partial x}u\right)^2 + \left(\frac{\partial I}{\partial y}v\right)^2 + 2\frac{\partial I}{\partial x}\frac{\partial I}{\partial y}uv\right]$$

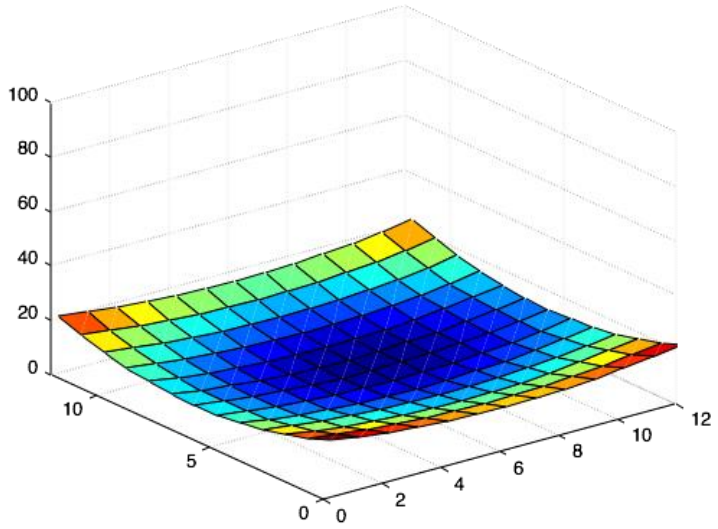
$$= \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} \sum_{p\in P}\left(\frac{\partial I}{\partial x}\right)^2 & \sum_{p\in P}\frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \sum_{p\in P}\frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \sum_{p\in P}\left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

Covariance matrix

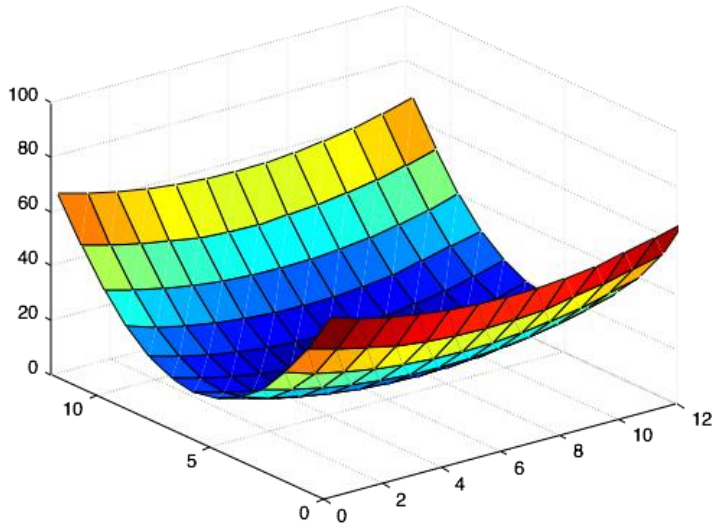
where $w(x,y) = 1$

Harris Corner Detection: ② **Covariance Matrix**

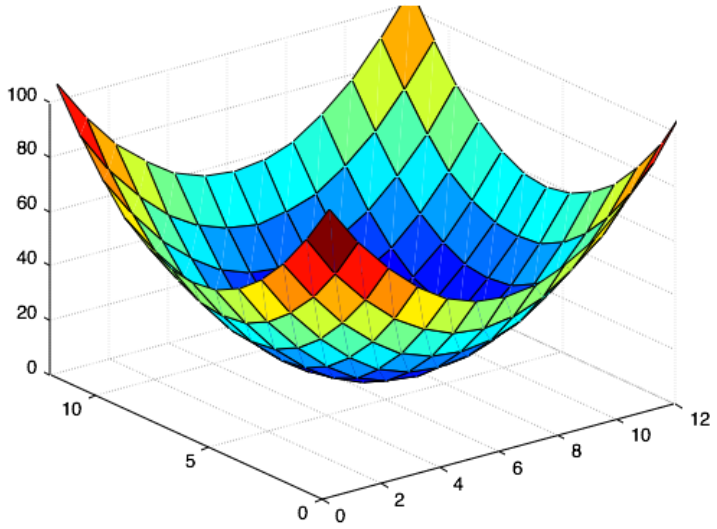
- Visualization of a quadratic error surface, $E(u, v)$



Flat



Edge



Corner

Harris Corner Detection: ③ **Eigenvalues & Eigenvectors**

- Compute **eigenvalues** and **eigenvectors**

- Given a square matrix $\mathbf{A}$, a scalar λ is called an eigenvalue of $\mathbf{A}$ if there exists a non-zero vector $\mathbf{v}$ that satisfies:

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$$

- The vector $\mathbf{v}$ is called an eigenvector for $\mathbf{A}$ corresponding to the eigenvalue λ
- The eigenvalues of $\mathbf{A}$ can be obtained by solving

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{v} = 0 \quad \longrightarrow \quad \det(\mathbf{A} - \lambda\mathbf{I}) = 0$$

Example: **Eigenvalues & Eigenvectors**

- λ is eigenvalue of $\mathbf{A}$ iff $det(\lambda \mathbf{I} - \mathbf{A}) = 0$
- Find the eigenvalue of a given matrix $\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 4 & 3 \end{bmatrix}$

$$\rightarrow det\left(\lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 2 \\ 4 & 3 \end{bmatrix}\right) = det\left(\begin{bmatrix} \lambda - 1 & -2 \\ -4 & \lambda - 3 \end{bmatrix}\right) = 0$$

$$\rightarrow \begin{vmatrix} \lambda - 1 & -2 \\ -4 & \lambda - 3 \end{vmatrix} = (\lambda - 1)(\lambda - 3) - 8 = \lambda^2 - 4\lambda - 5 = (\lambda - 5)(\lambda + 1) = 0$$

$$\rightarrow \therefore \lambda = 5, -1$$

Example: **Eigenvalues & Eigenvectors**

- Eigenspace, $E_\lambda = N(\lambda \mathbf{I} - \mathbf{A})$
- Find the eigenvector and eigenspace of a given matrix $\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 4 & 3 \end{bmatrix}$

$$\rightarrow \begin{cases} \lambda = 5, & E_{\lambda=5} = N\left(\begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} - \begin{bmatrix} 1 & 2 \\ 4 & 3 \end{bmatrix}\right) = N\left(\begin{bmatrix} 4 & -2 \\ -4 & 2 \end{bmatrix}\right) \\ \lambda = -1, & E_{\lambda=-1} = N\left(\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} - \begin{bmatrix} 1 & 2 \\ 4 & 3 \end{bmatrix}\right) = N\left(\begin{bmatrix} -2 & -2 \\ -4 & -4 \end{bmatrix}\right) \end{cases}$$

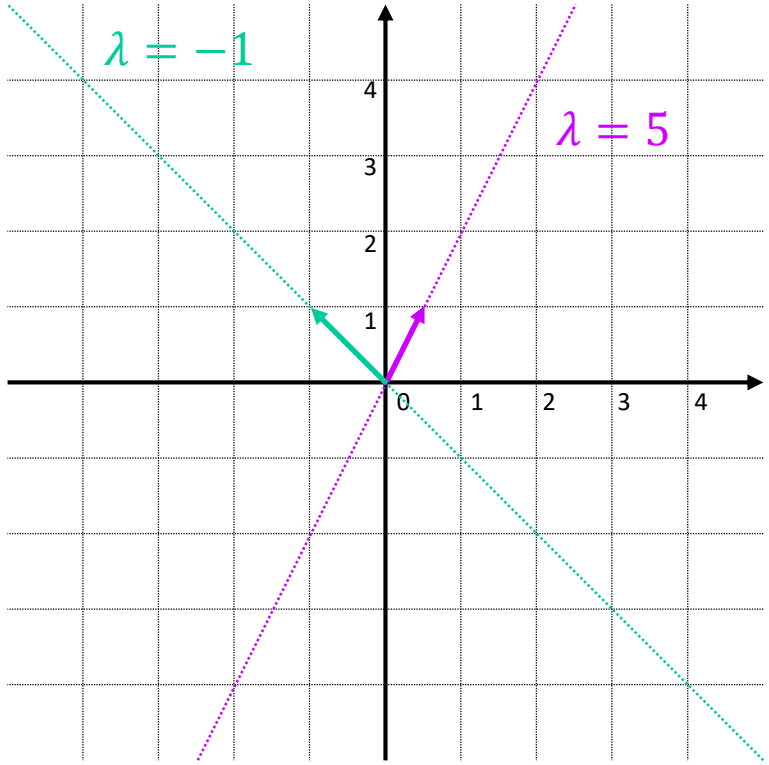
$$\rightarrow \begin{cases} \begin{bmatrix} 4 & -2 \\ -4 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \mathbf{0} \rightarrow \begin{bmatrix} 1 & -\frac{1}{2} \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \rightarrow x_1 - \frac{1}{2}x_2 = 0 \\ \begin{bmatrix} -2 & -2 \\ -4 & -4 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \mathbf{0} \rightarrow \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \rightarrow x_1 + x_2 = 0 \end{cases}$$

Example: **Eigenvalues & Eigenvectors**

- Eigenspace, $E_\lambda = N(\lambda \mathbf{I} - \mathbf{A})$
- Find the eigenvector and eigenspace of a given matrix $\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 4 & 3 \end{bmatrix}$

$$\therefore\ E_{\lambda=5} = \left\{ \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = t \begin{bmatrix} \frac{1}{2} \\ 1 \end{bmatrix} \mid t \in \mathbb{R} \right\} = Span\left(\begin{bmatrix} \frac{1}{2} \\ 1 \end{bmatrix} \right)$$

$$E_{\lambda=-1} = \left\{ \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = t \begin{bmatrix} -1 \\ 1 \end{bmatrix} \mid t \in \mathbb{R} \right\} = Span\left(\begin{bmatrix} -1 \\ 1 \end{bmatrix} \right)$$



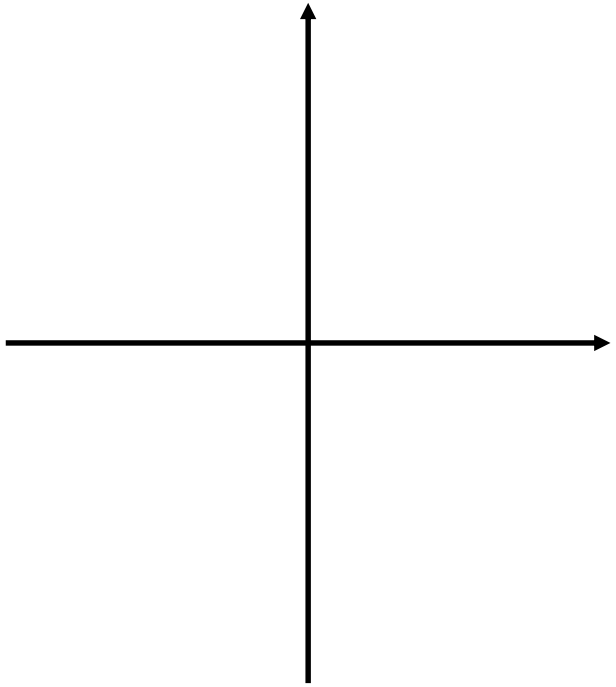
- Diagonalization of $\mathbf{A}$

$$\begin{bmatrix} 1 & 2 \\ 4 & 3 \end{bmatrix} = \begin{bmatrix} 1/2 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 5 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1/2 & -1 \\ 1 & 1 \end{bmatrix}^{-1}$$

Harris Corner Detection: ③ **Eigenvalues & Eigenvectors**

- **Visualization** as an **Ellipse**

$$\mathbf{M} = \begin{bmatrix} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\ \sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{bmatrix} = [\mathbf{v}_1 \quad \mathbf{v}_2] \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} [\mathbf{v}_1 \quad \mathbf{v}_2]^{-1}$$



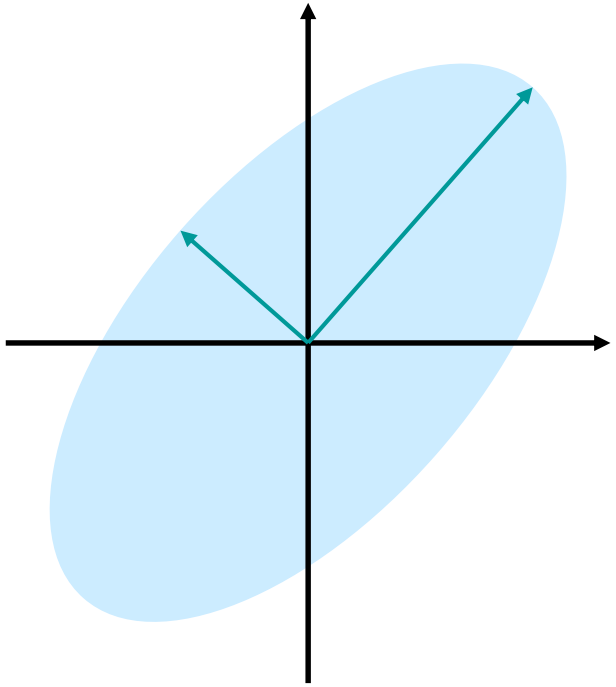
Harris Corner Detection: ③ **Eigenvalues & Eigenvectors**

- **Visualization** as an **Ellipse**

— **Eigenvectors** determines the **orientation of the ellipse**

$$\mathbf{M} = \begin{bmatrix} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\ \sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{bmatrix} = [\mathbf{v}_1 \quad \mathbf{v}_2] \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} [\mathbf{v}_1 \quad \mathbf{v}_2]^{-1}$$

Eigenvectors



Harris Corner Detection: ③ **Eigenvalues & Eigenvectors**

- **Visualization** as an **Ellipse**

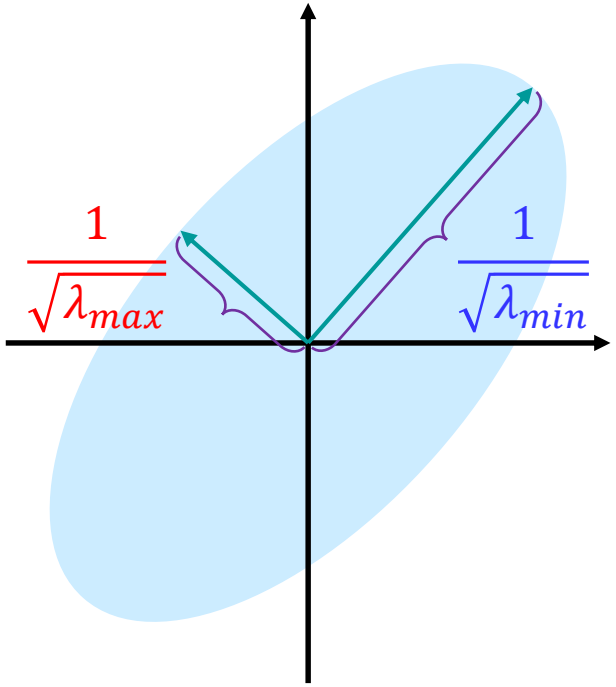
— **Eigenvectors** determines the **orientation of the ellipse**

— **Eigenvalues** determines the axis **lengths of the ellipse**

  - $\frac{1}{\sqrt{\lambda_{max}}}$: The direction of the fastest change
  - $\frac{1}{\sqrt{\lambda_{min}}}$: The direction of the slowest change

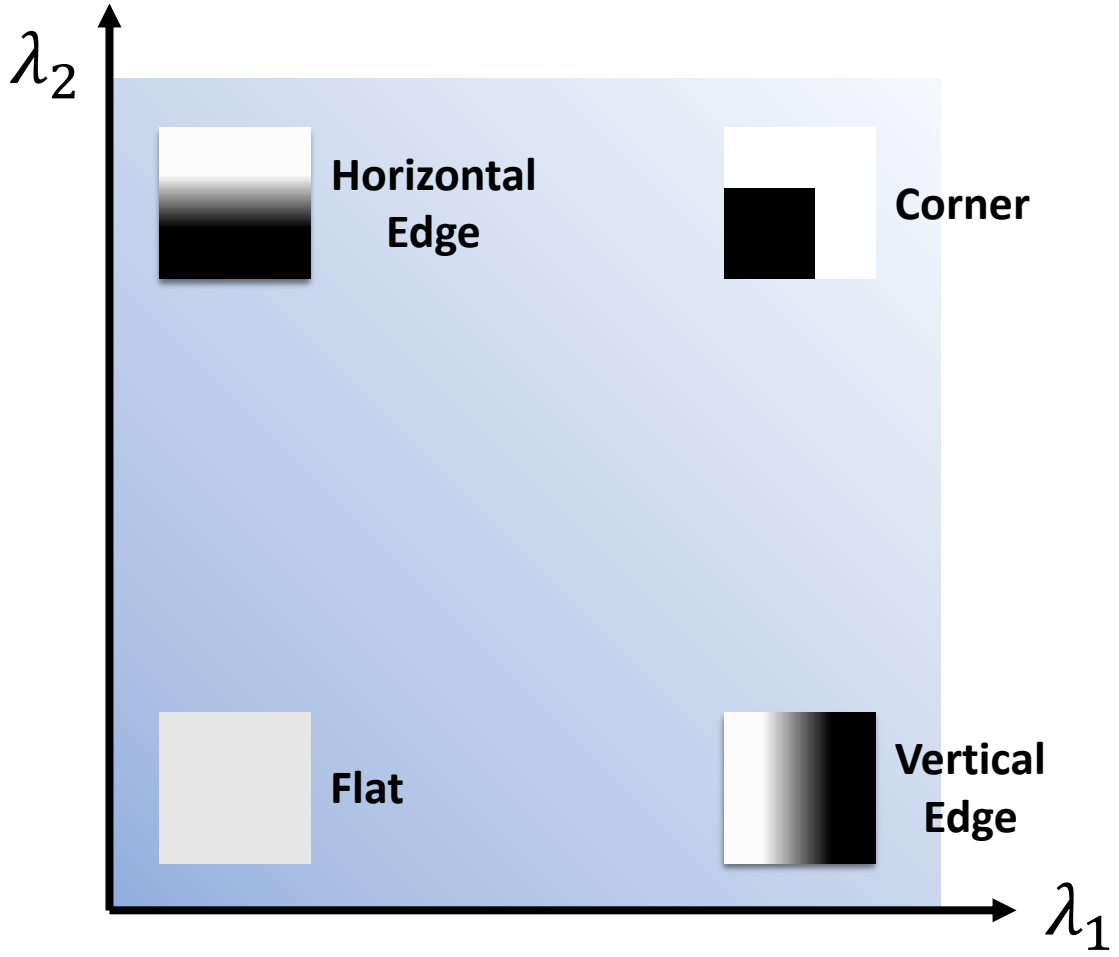
$$\mathbf{M} = \begin{bmatrix} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\ \sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{bmatrix} = [\mathbf{v}_1 \quad \mathbf{v}_2] \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} [\mathbf{v}_1 \quad \mathbf{v}_2]^{-1}$$

Eigenvectors Eigenvalues



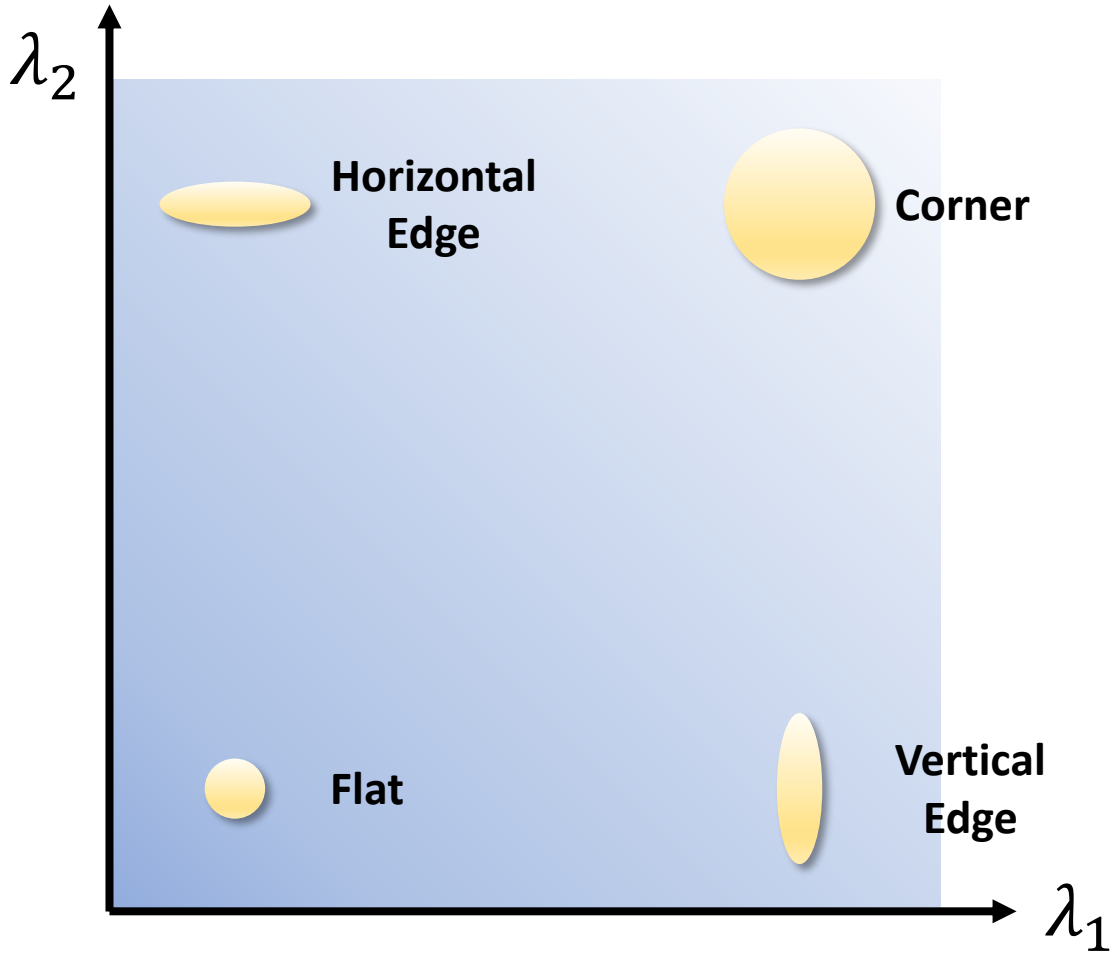
Harris Corner Detection: ③ **Eigenvalues & Eigenvectors**

- **Classification of Image Points** using Eigenvalues of **C**



Harris Corner Detection: ③ **Eigenvalues & Eigenvectors**

- **Classification of Image Points** using Eigenvalues of **C**



Harris Corner Detection: ④ **Eigenvalue Thresholding**

- Use **threshold on eigenvalues** to detect corners

- **Cornerness:**

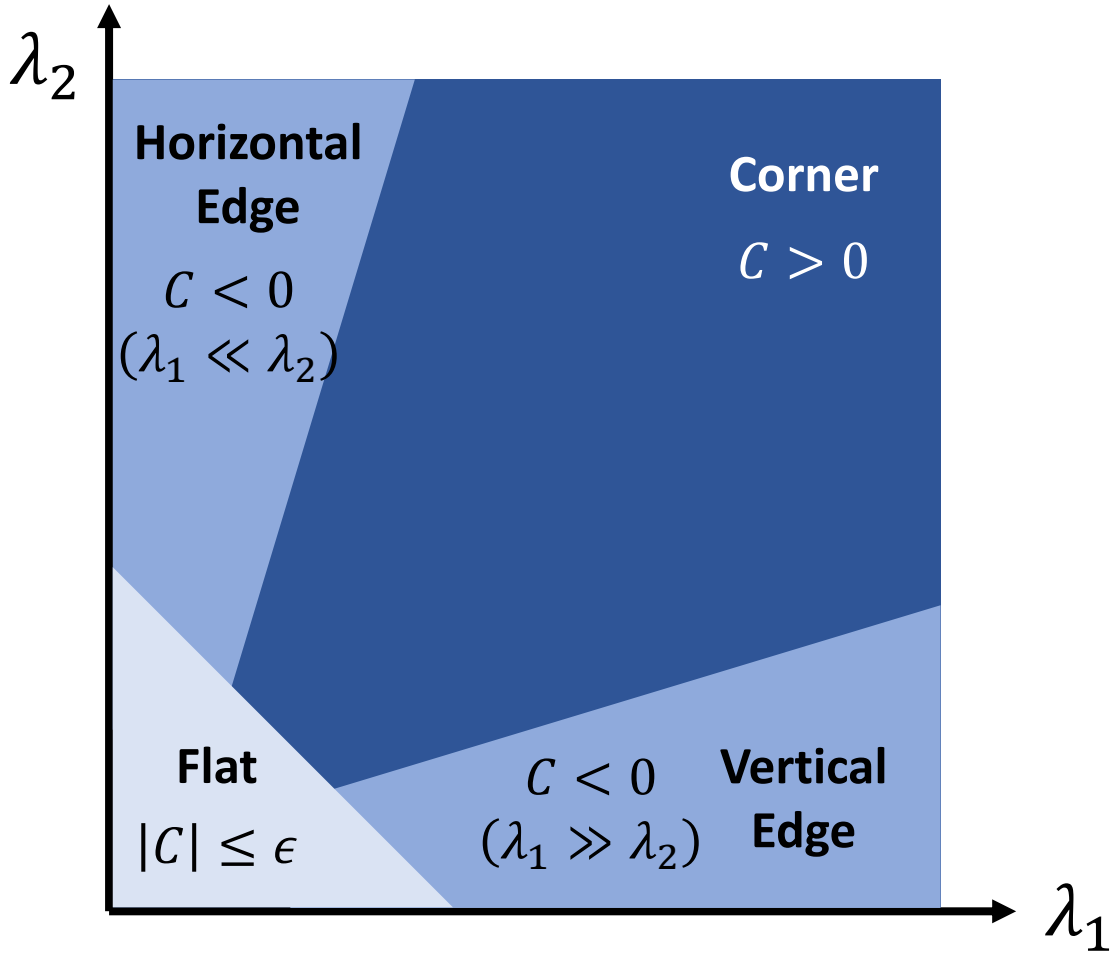
$$C = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$
$$= \det(\mathbf{M}) - \alpha \cdot \mathrm{tr}(\mathbf{M})^2$$

$0.04 \leq \alpha \leq 0.06$

— Flat region: $|C| \leq \epsilon$

— Edge: $C < 0$

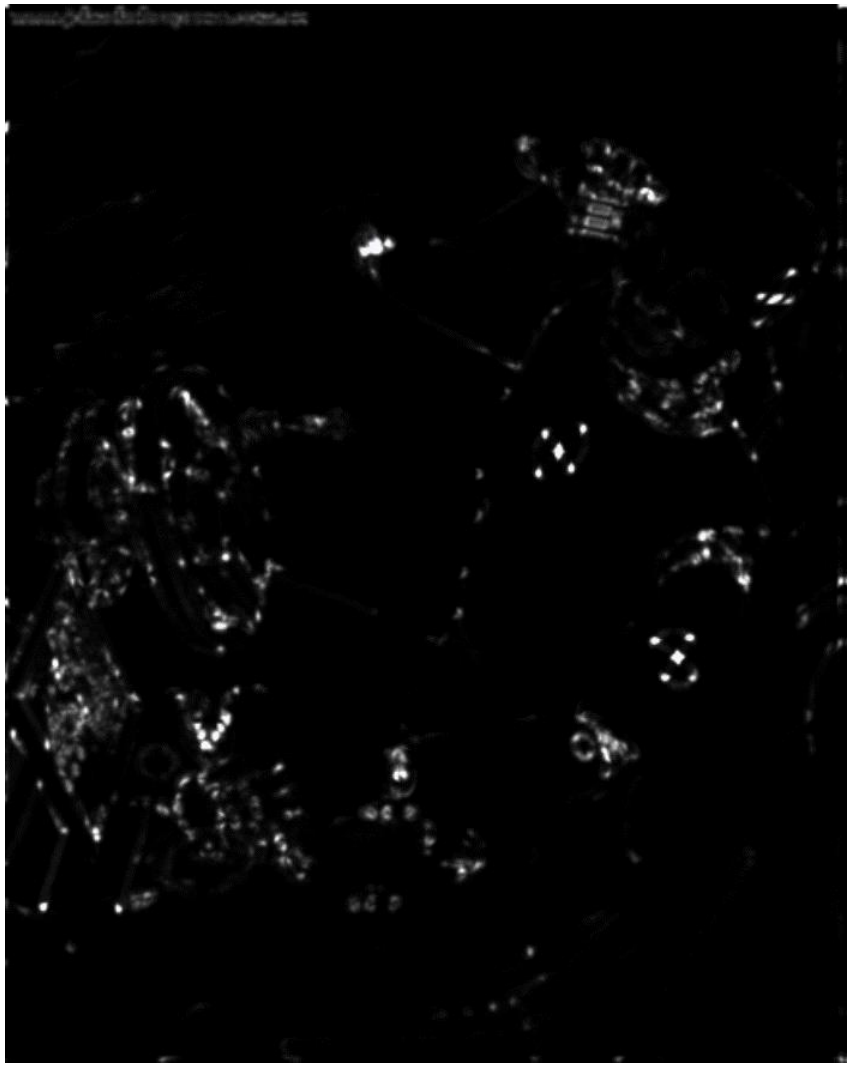
— Corner: $C > 0$



Example: Harris Corner Detection

Example: Harris Corner Detection

Example: Harris Corner Detection



Corner response map

Corner detection result ($\sigma = 1$)

Summary: **Harris Corner Detection**

- **Algorithm:**

① Compute **image gradient** over small region

② Compute the **covariance matrix**

③ Compute **eigenvectors** and **eigenvalues**

④ Use **threshold on eigenvalues** to detect corners

- If **both** eigenvalues are **big**, then it has a **corner**
- If **both** eigenvalues are **small**, then it is a **flat** region
- If **one** eigenvalue is **much larger than the other**, then it has an **edge**