Computer Vision (CSE3010)

Dr. Susant Kumar Panigrahi
Assistant Professor
School of Electrical & Electronics Engineering

VIT BHOPAL

Module-1 Syllabus

Digital Image Formation And Low Level Processing:

- Overview and State-of-the-art, Fundamentals of Image Formation, Transformation: Orthogonal, Euclidean, Affine, Projective, Fourier Transform,
- Convolution and Filtering, Image Enhancement, Restoration, Histogram Processing.

Module-2 Syllabus

Depth Estimation And Multi-Camera Views:

Depth Estimation and Multi-Camera Views: Perspective, Binocular Stereopsis: Camera and Epipolar Geometry; Homography, Rectification, DLT, RANSAC, 3-D reconstruction framework; Auto-calibration. apparel.

Module-3 Syllabus

Feature Extraction And Image Segmentation:

- **Feature Extraction**: Edges - Canny, LOG, DOG; Line detectors (Hough Transform), Corners - Harris and Hessian Affine, Orientation Histogram, SIFT, SURF, HOG, GLOH, Scale-Space Analysis- Image Pyramids and Gaussian derivative filters, Gabor Filters and DWT.
- **Image Segmentation:** Region Growing, Edge Based approaches to segmentation, Graph-Cut, Mean-Shift, MRFs, Texture Segmentation; Object detection.

Module-4 Syllabus

Pattern Analysis And Motion Analysis:

- **Pattern Analysis**: Clustering: K-Means, K-Medoids, Mixture of Gaussians, Classification: Discriminant Function, Supervised, Un-supervised, Semi-supervised; Classifiers: Bayes, KNN, ANN models;
- **Dimensionality Reduction**: PCA, LDA, ICA; Non-parametric methods. Motion Analysis: Background Subtraction and Modelling, Optical Flow, KLT, Spatio-Temporal Analysis, Dynamic Stereo; Motion parameter estimation.

Module-5 Syllabus

Shape From X:

Light at Surfaces; Phong Model; Reflectance Map;

Albedo estimation; Photometric Stereo; Use of Surface Smoothness

Constraint; Shape from Texture, color, motion and edges.

Guest Lecture on Contemporary Topics

Text Books

1. Richard Szeliski, Computer Vision: Algorithms and Applications, Springer-Verlag London Limited 2011.
2. Computer Vision: A Modern Approach, D. A. Forsyth, J. Ponce, Pearson Education, 2003.

Reference Book(s):

1. R.C. Gonzalez and R.E. Woods, Digital Image Processing, Addison- Wesley, 1992.
2. Richard Hartley and Andrew Zisserman, Multiple View Geometry in Computer Vision, Second Edition, Cambridge University Press, March 2004.
3. K. Fukunaga; Introduction to Statistical Pattern Recognition, Second Edition, Academic Press, Morgan Kaufmann, 1990.

Required Tools/Software/IDLE:

1. Python/jupyter-notebook/google-colab
2. OpenCV
3. MATLAB

Indicative List of Experiments:

1. Implement image preprocessing and Edge
2. Implement camera calibration methods
3. Implement Projection
4. Determine depth map from Stereo pair
5. Construct 3D model from Stereo pair
6. Implement Segmentation methods
7. Construct 3D model from defocus image
8. Construct 3D model from Images
9. Implement optical flow method
10. Implement object detection and tracking from video
11. Face detection and Recognition
12. Object detection from dynamic Background for Surveillance
13. Content based video retrieval
14. Construct 3D model from single image

Computer Vision
Unit – 03
Feature Extraction: Edge Detection

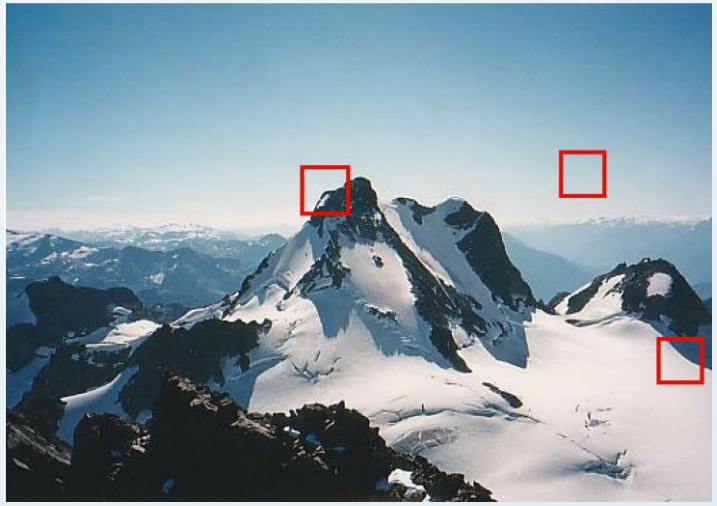
Standing on the shoulder of Giants: Ref: Few Slides borrowed from:

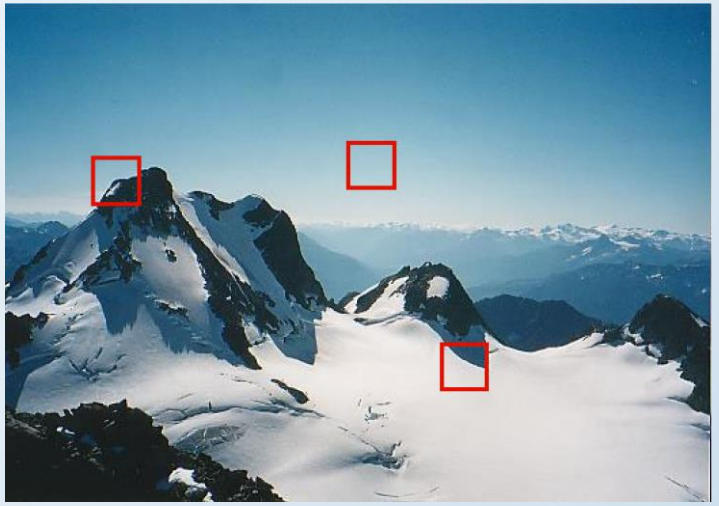
1. Prof. Shree Nayar, *First Principles of Computer Vision is a lecture series*.
2. Prof. Mubarak Shah, *Computer Vision Video Lectures.*

Image Features

- ✓ Features can be described as **interest points** in an image.
- ✓ Features are interesting part of a particular image that may aide us in uniquely identifying the image to perform several necessary computer vison tasks.

(a)





(b)

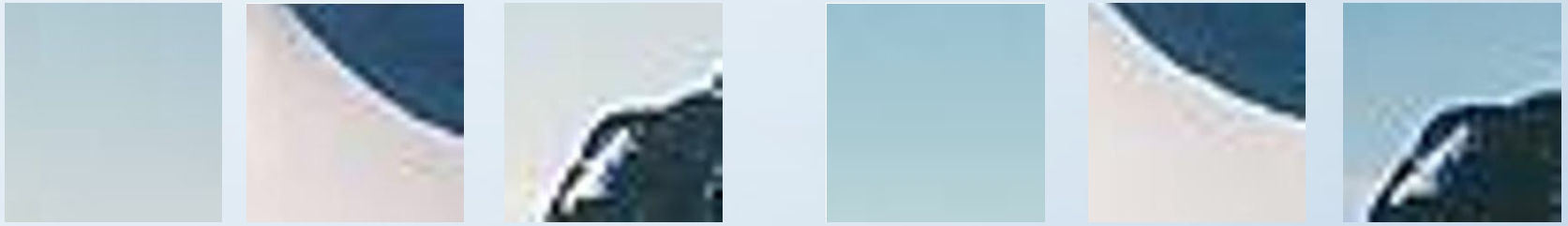


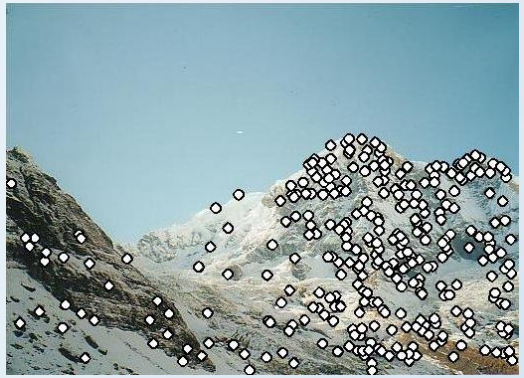
Image Pair. Few patches extracted from each image.
Notice some patches are very distinctive and unique compared to others.
The unique patches localized in an image are called interest point (features).

Properties of interest point

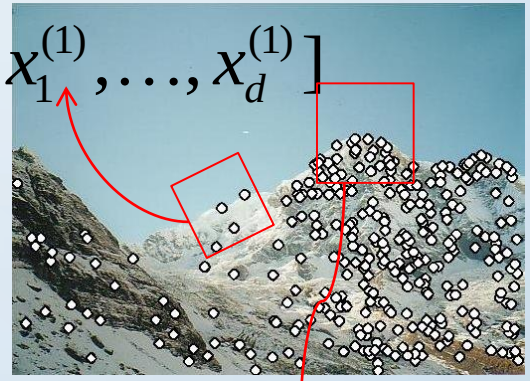
- ✓ Feature must be
  - Distinctive and unique (different from the near neighbourhood)
  - Repeatable: Interest points can be found in multiple images using same operations.
  - Local: Occupy small portion of a larger image.
  - Enough: High in quantity to represent the whole image
  - Efficient: Less computational complexity to detect points of interest. As in many applications the point of interest detector used as an pre-processing step.

Three fundamental components of Local features

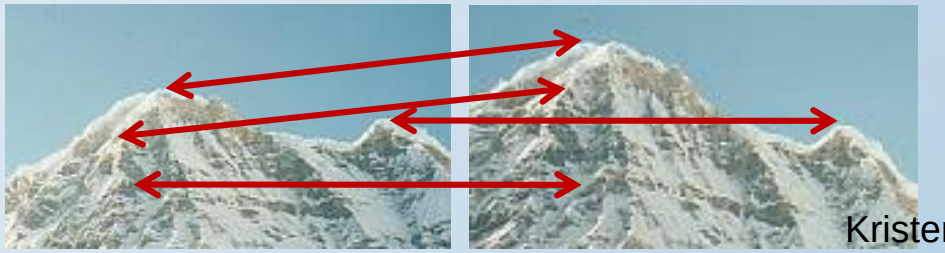
1) **Detection**: Identify the local features (**interest points**) in an image.



2) **Description:** Extracting the multi-dimensional feature vectors in the neighborhood of the interest point.



3) **Matching**: Measuring the similarity between two feature vectors (descriptors) to match two view points.



Kristen Grauman

Where can we use it?

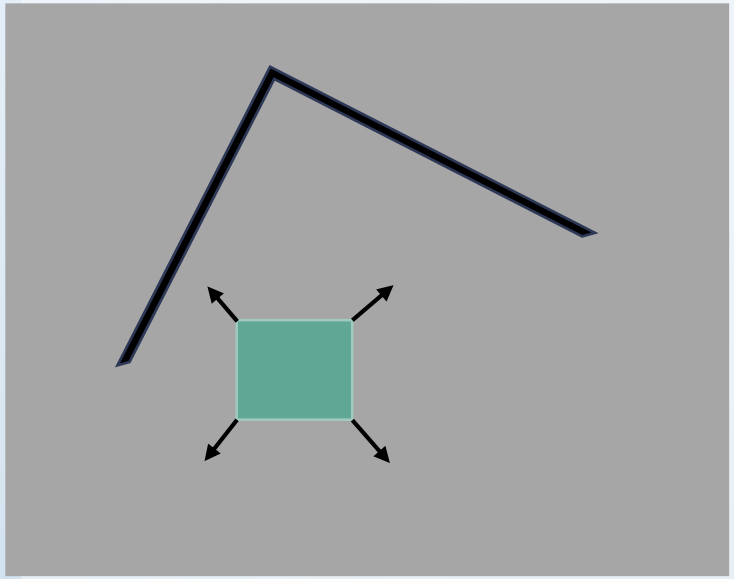
- Automatic object tracing
- Motion based segmentation
- Recognition
- 3D Object recognition
- Robot Navigation
- Stereo Calibration
  - ✓ Estimation of fundamental matrix
- Image retrieval and indexing

Extracting features

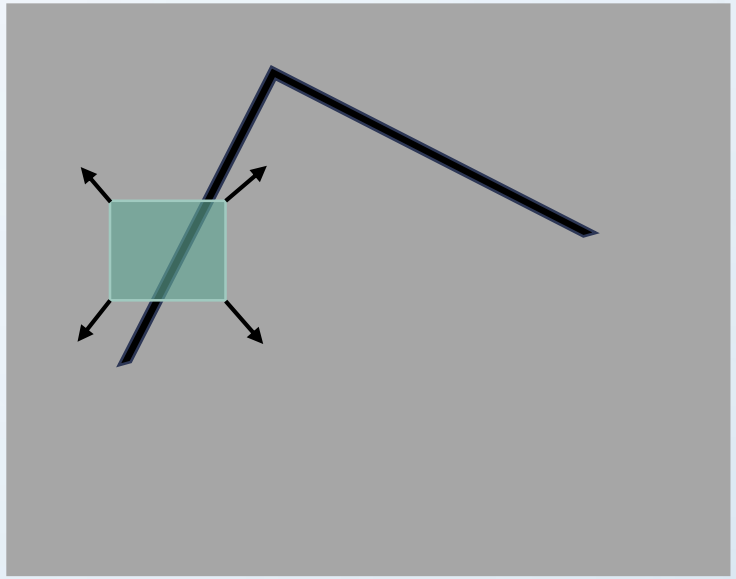
- Reparative texture-less patches (flat or homogenous regions of an image) are very difficult to detect efficiently.
- Regions with large change in intensity (gradient) in one direction are little easier to detect (edges).
- Point of interest in an image could easily be extracted if gradients changes (significantly) at least in two directions (corners).

Pic Courtesy: The Cityscapes Dataset: https://www.cityscapes-dataset.com/

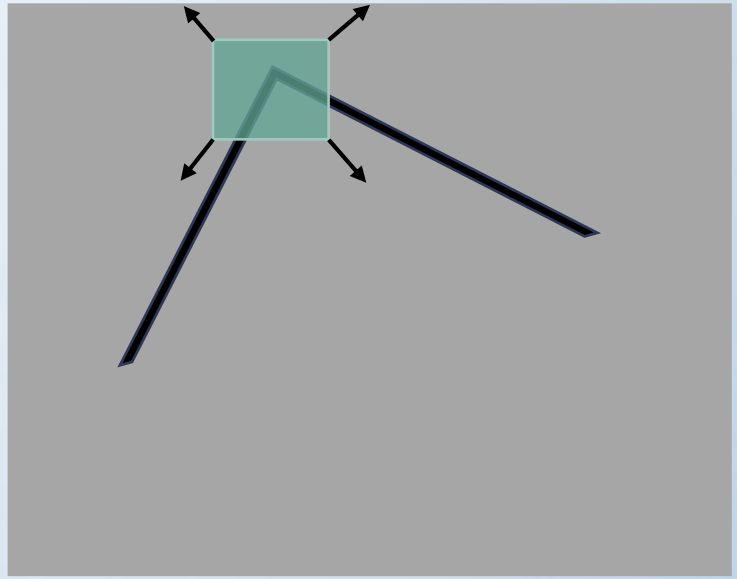
Feature Detector: Basic Idea



‘Flat/Homogenous region’ no changes in all the directions.



‘Edges/Gradient in one direction’ no changes along the edge direction.



‘Corner’ significant change in all directions.

Edge Detection: An Overview

- Edge detections is a technique that converts a 2D image into a set of points, where image intensity changes rapidly.

- Pixel at which the intensity of an image change abruptly.

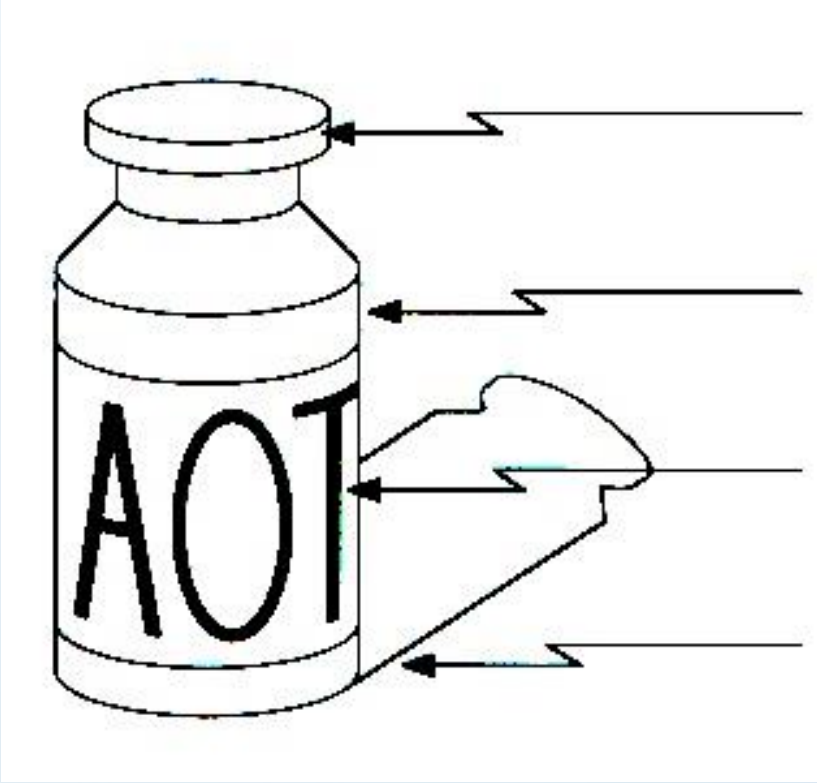
Topics:

1. What is an Edge?
2. Edge Detection using Gradients
3. Edge Detection using Laplacian
4. DOG and LOG
5. Canny Edge Detection

What Causes Edges in Image?

- Edges conveys most of the visual information that are preserved by human eyes.

- Rapid change in image intensity are caused by various physical phenomena.



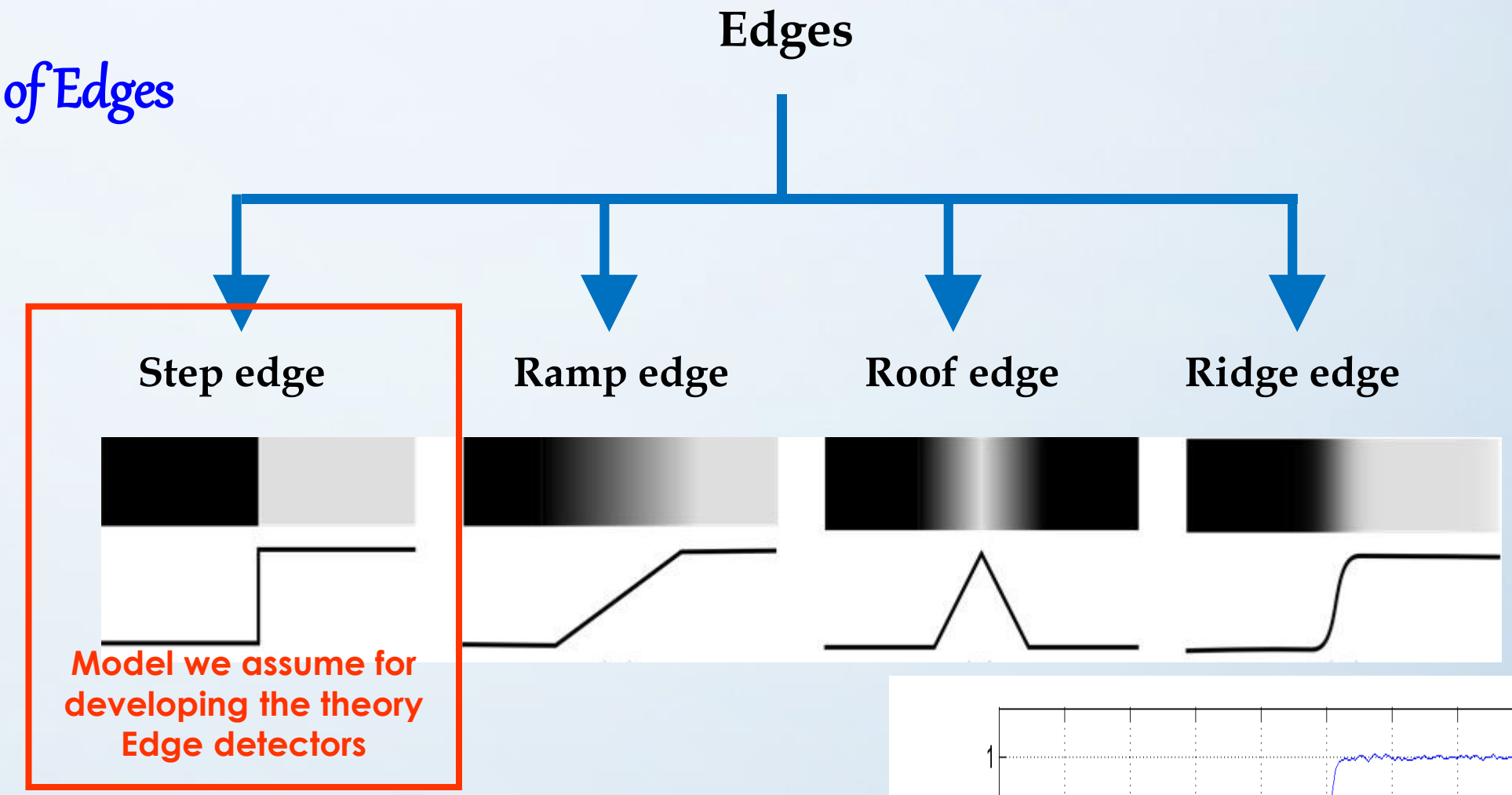
Surface normal discontinuity

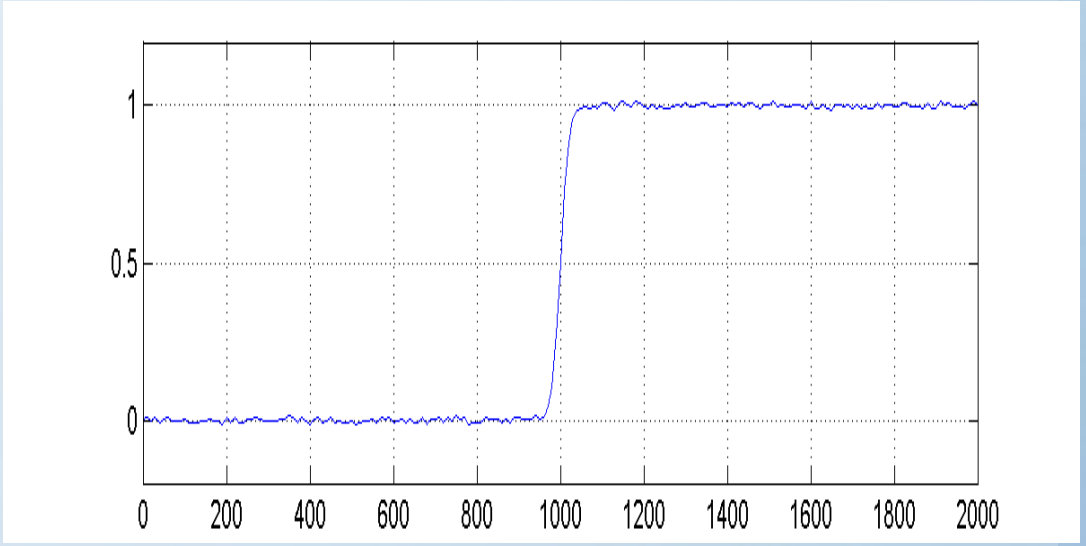
Depth discontinuity

Surface Reflectance/Color discontinuity

Illumination discontinuity

Types of Edges



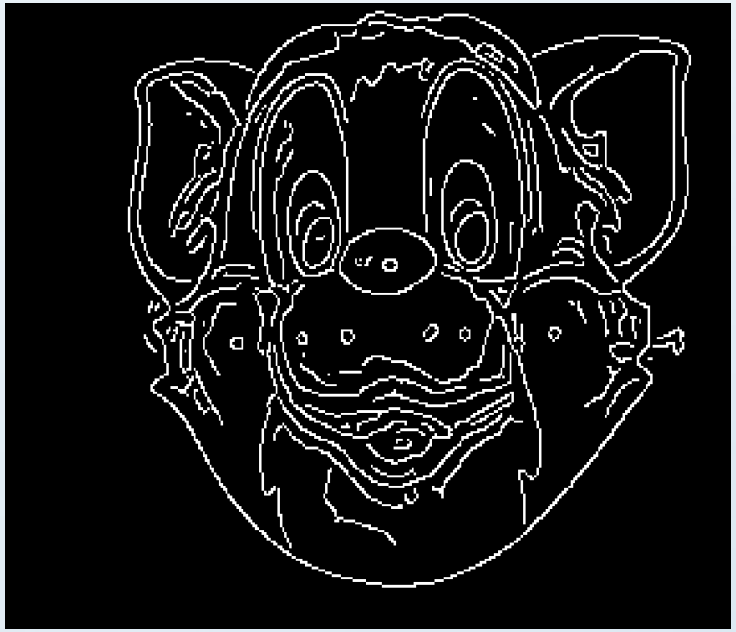


Real Edges in an image far distorted that the ideal models: Noisy Images and Discrete Images.

Edge Detector

We want an **Edge Operator** that produce:

- Edge Position
- Edge Magnitude (Strength)
- Edge Orientation (Direction)

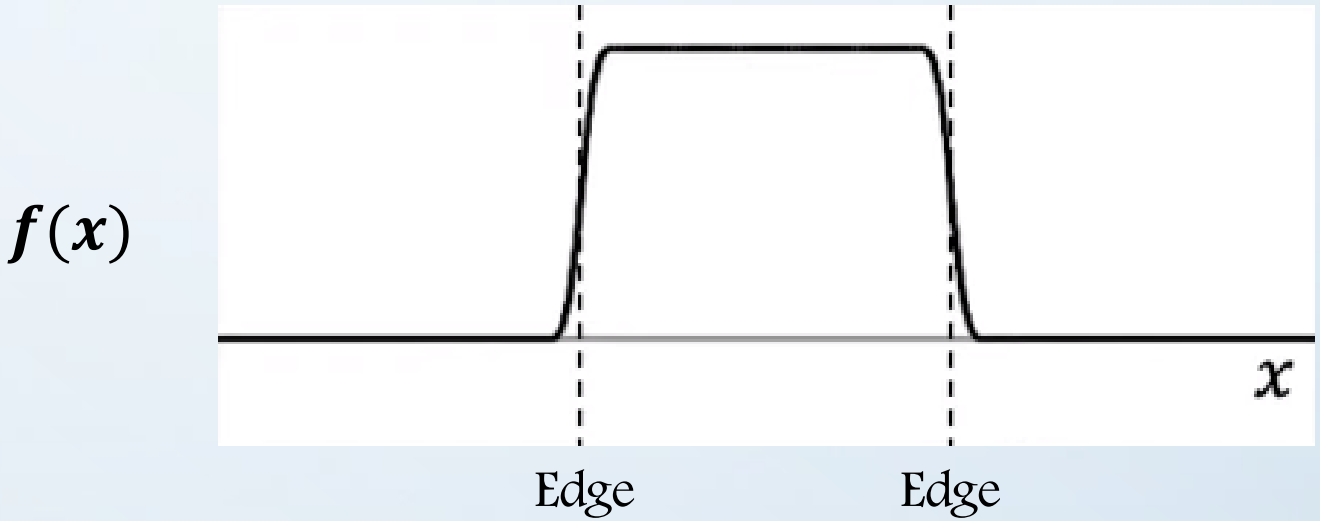


Performance Requirement:

- High Detection Rate
- Good Localization
- Low Noise Sensitivity

Edge Detection using Gradients

Edge is a **rapid change** in intensity in a small region.



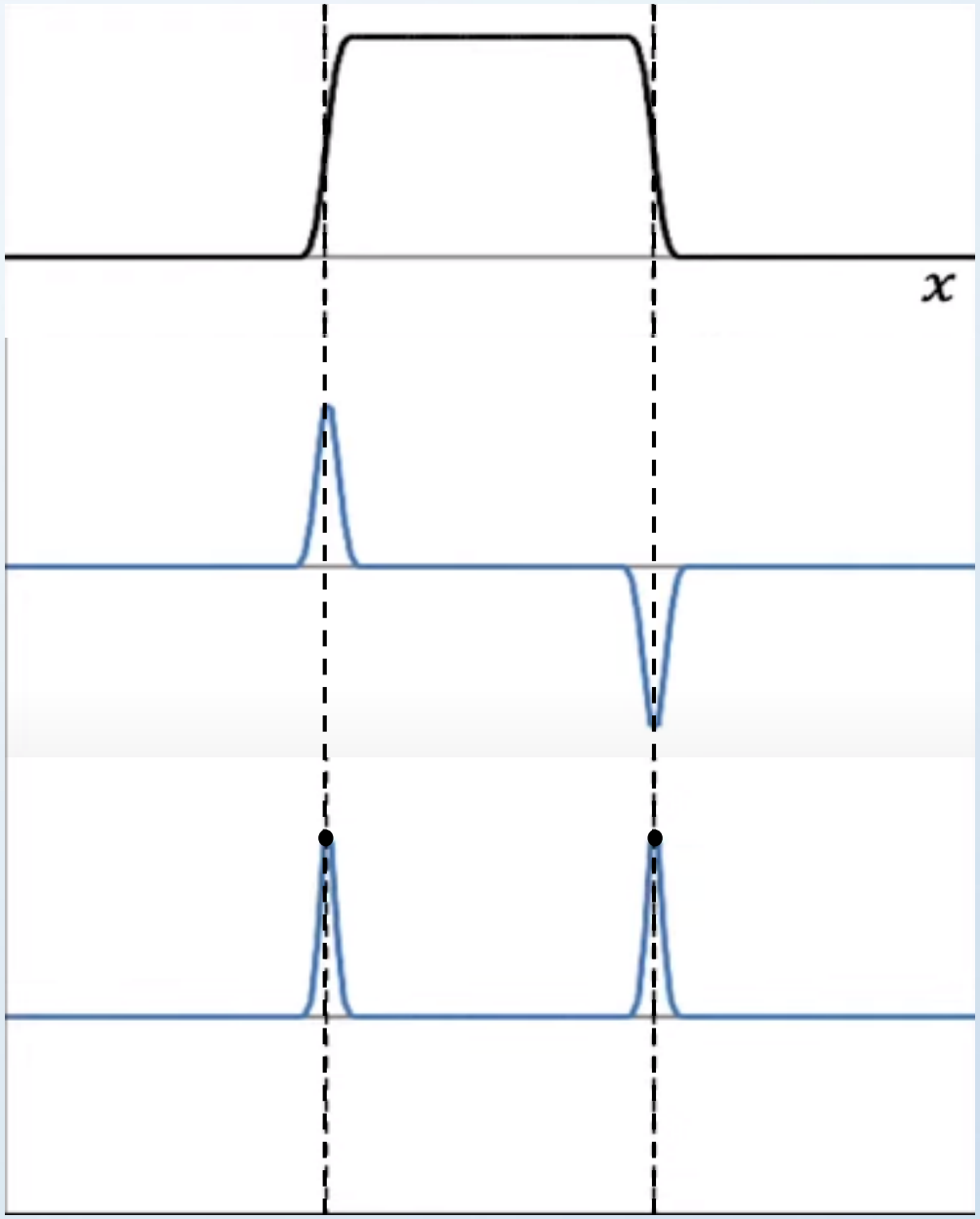
Basic Calculus: **Derivative** of a continuous function represent the amount of change in a function.

Edge Detection using 1st Derivative

$f(x)$

First Derivative: $\frac{\partial f}{\partial x}$

First Derivative Absolute Value: $\left|\frac{\partial f}{\partial x}\right|$



Local Extrema
Indicate Edges

Local Maxima
Indicate Edges

2D Edge Detection

$I(x, y)$

Edge

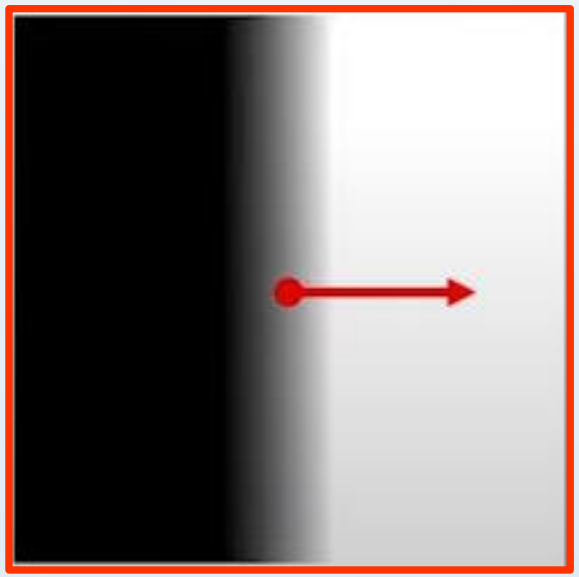
Edge

Basic Calculus: Partial Derivative of a 2D continuous function represents the amount of change along each directions.

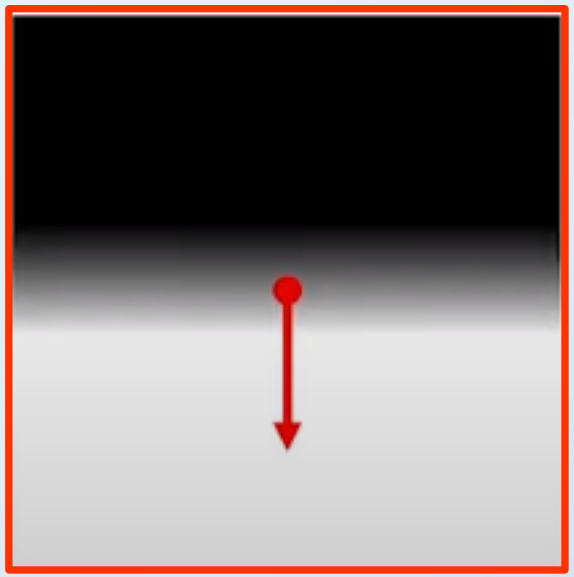
Gradient Operator (∇)

Gradient (Partial Derivative) *represent the direction of most rapid change in intensity value of a 2D image.*

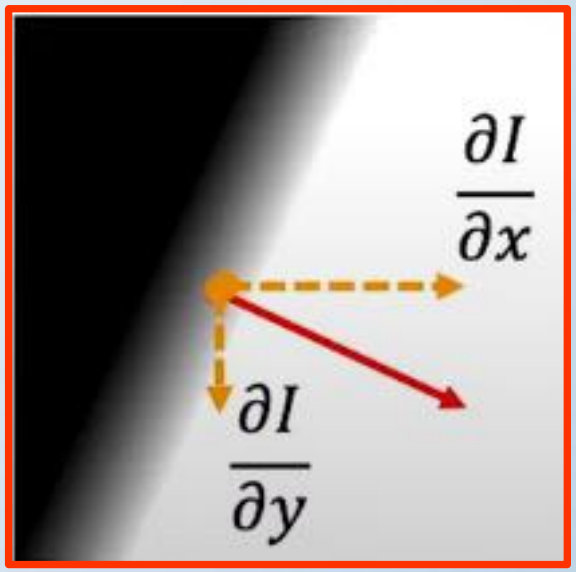
$$\nabla I = \left[\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right]$$



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



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



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

Gradient (∇) as Edge Detector

Gradient Magnitude:

$$S = \|\nabla I\| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$$

Gradient Orientation:

$$\theta = \tan^{-1}\left(\frac{\partial I}{\partial y} / \frac{\partial I}{\partial x}\right)$$

Gradient Operator: It is a vector quantity.

Discrete Gradient (∇) Operator

Finite difference approximation:

$$\frac{\partial I}{\partial x} \approx \frac{1}{2\epsilon}\left(\left(I_{i+1,j+1} - I_{i,j+1}\right) - \left(I_{i+1,j} - I_{i,j}\right)\right)$$

$$\frac{\partial I}{\partial y} \approx \frac{1}{2\epsilon}\left(\left(I_{i+1,j+1} - I_{i+1,j}\right) - \left(I_{i,j+1} - I_{i,j}\right)\right)$$

$I_{i,j+1}$	$I_{i,j+1}$
$I_{i,j+1}$	$I_{i,j+1}$

ϵ

Can be implemented as convolution!

$$\frac{\partial I}{\partial x} \approx \frac{1}{2\epsilon}$$

-1	1
-1	1

$$\frac{\partial I}{\partial y} \approx \frac{1}{2\epsilon}$$

1	1
-1	-1

Comparing Discrete Gradient (∇) Operator

Gradients	Roberts	Prewitt	Sobel (3×3)	Sobel (5×5)
$\frac{\partial I}{\partial x}$	$\begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -2 & 0 & 2 & 1 \\ -2 & -3 & 0 & 3 & 2 \\ -3 & -5 & 0 & 5 & 3 \\ -2 & -3 & 0 & 3 & 2 \\ -1 & -2 & 0 & 2 & 1 \end{bmatrix}$
$\frac{\partial I}{\partial y}$	$\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 3 & 5 & 3 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ -2 & -3 & -5 & -3 & -2 \\ -1 & -2 & -3 & -2 & -1 \end{bmatrix}$

Comparing Discrete Gradient (∇) Operator

Good Localization

Noise Sensitivity

Poor Detection

Good Localization

Less Noise Sensitivity

Good Detection

Gradient (∇) using Sobel (3×3) Operator

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

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

Gradient Magnitude

Edge Thesholding

Standard: (Single Threshold T)

$$\|\nabla I(x,y)\| < T \qquad \text{Definitely not an Edge}$$

$$\|\nabla I(x,y)\| \geq T \qquad \text{Definitely an Edge}$$

Hysteresis Based: (Two Thresholds $T_0 < T_1$)

$$\|\nabla I(x,y)\| < T_0 \qquad \text{Definitely not an Edge}$$

$$\|\nabla I(x,y)\| \geq T_1 \qquad \text{Definitely an Edge}$$

$$T_0 \leq \|\nabla I(x,y)\| \leq T_1 \qquad \text{Is an Edge if the neighboring pixel is definitely an edge}$$

Sobel Edge Detector

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

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

Gradient Magnitude

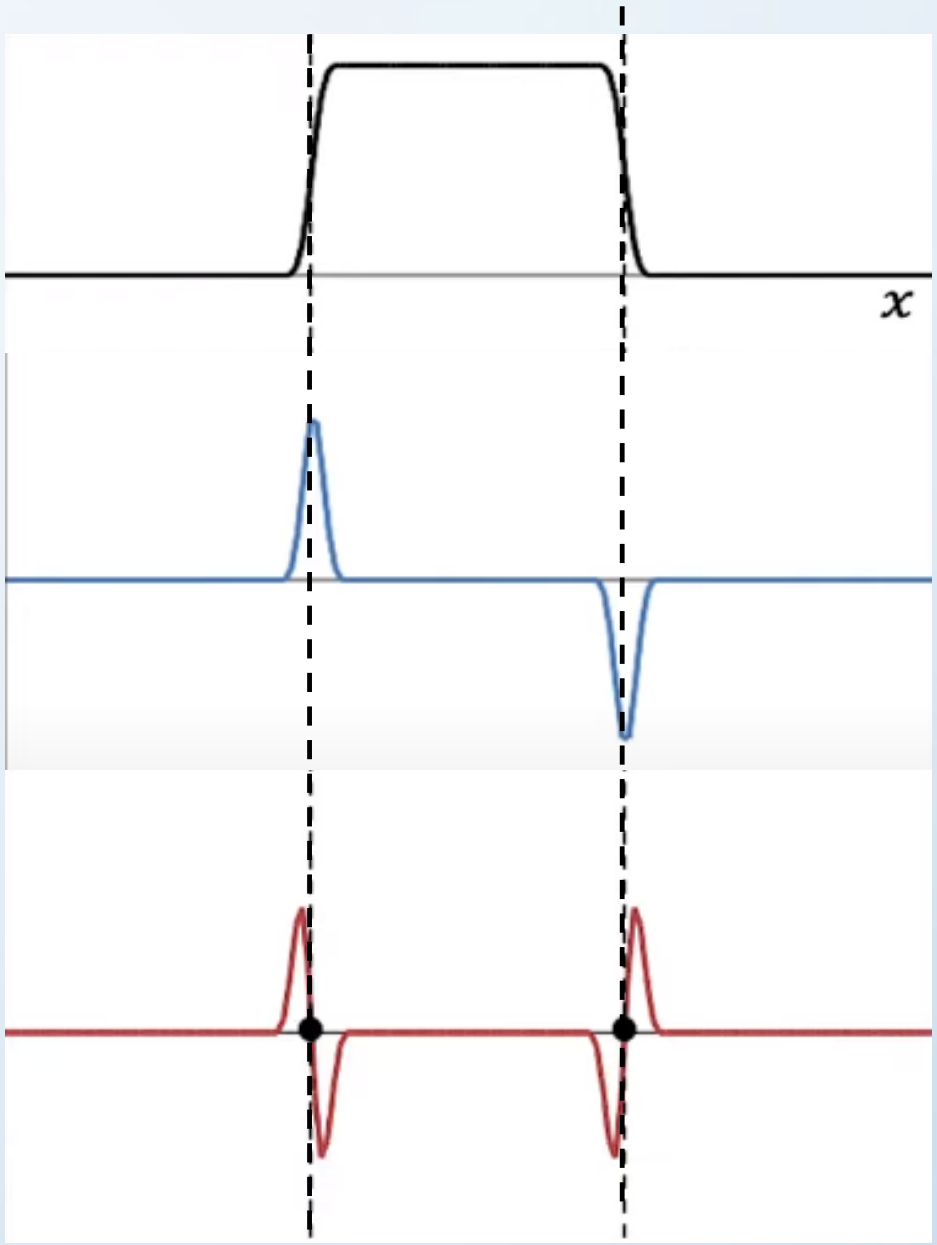
Thesholded Image: Edge Map

Edge Detection using 2nd Derivative

$f(x)$

First Derivative: $\frac{\partial f}{\partial x}$

First Derivative Absolute Value: $\frac{\partial^2 f}{\partial x^2}$



Local Extrema
Indicate Edges

Zero Crossing
Indicate Edges

Laplacian (∇^2) as Edge Detector

Laplacian: Sum of pure second derivative.

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

- Edges are "Zero Crossing" in Laplacian of image.
- Laplacian doesn't provide direction of an edge.
- It is a scalar quantity.

Discrete Laplacian (∇^2) Operator

Finite difference approximation:

$$\frac{\partial^2 I}{\partial x^2} \approx \frac{1}{\epsilon^2}\left(I_{i-1,j} - 2I_{i,j} + I_{i+1,j}\right)$$

$$\frac{\partial^2 I}{\partial y^2} \approx \frac{1}{\epsilon^2}\left(I_{i,j-1} - 2I_{i,j} + I_{i,j+1}\right)$$

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

$I_{i-1,j+1}$	$I_{i,j+1}$	$I_{i+1,j+1}$
$I_{i-1,j}$	$I_{i,j}$	$I_{i+1,j}$
$I_{i-1,j-1}$	$I_{i,j-1}$	$I_{i+1,j-1}$

ϵ

Can be implemented as convolution!

$$\nabla^2 \approx \frac{1}{\epsilon^2}$$

0	1	0
1	-4	1
0	1	0

More Accurate

$$\nabla^2 \approx \frac{1}{6\epsilon^2}$$

1	4	1
4	-20	4
1	4	1

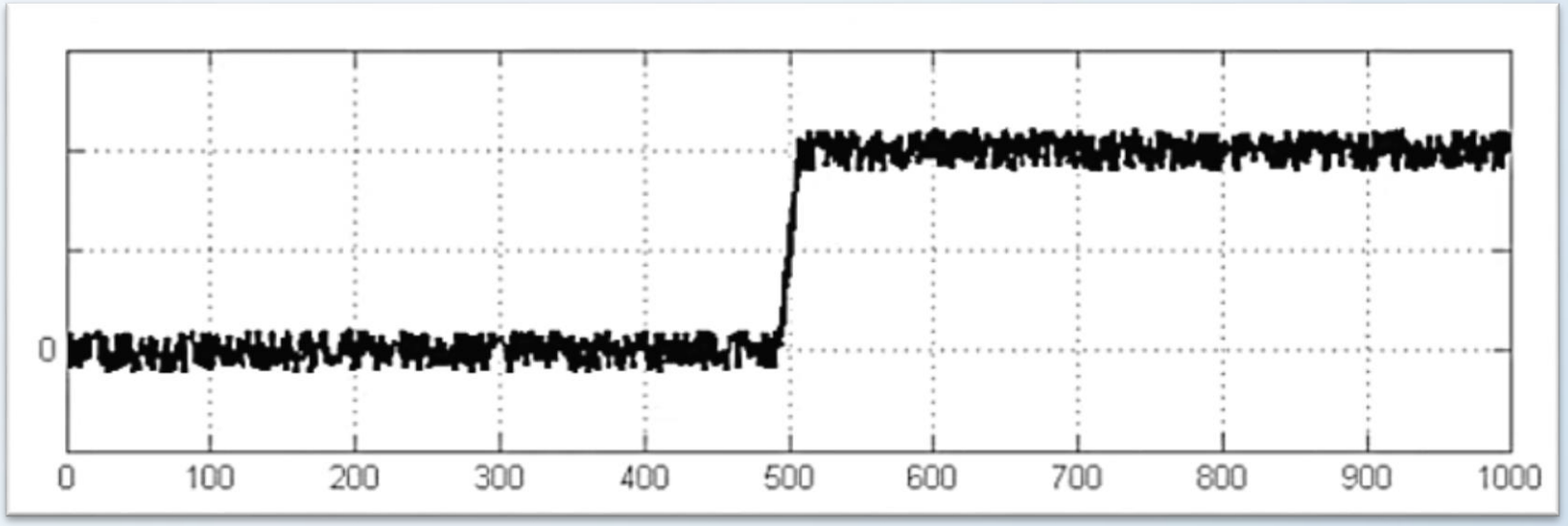
Laplacian (∇^2) Edge Detector

Laplacian
0 maps to 128

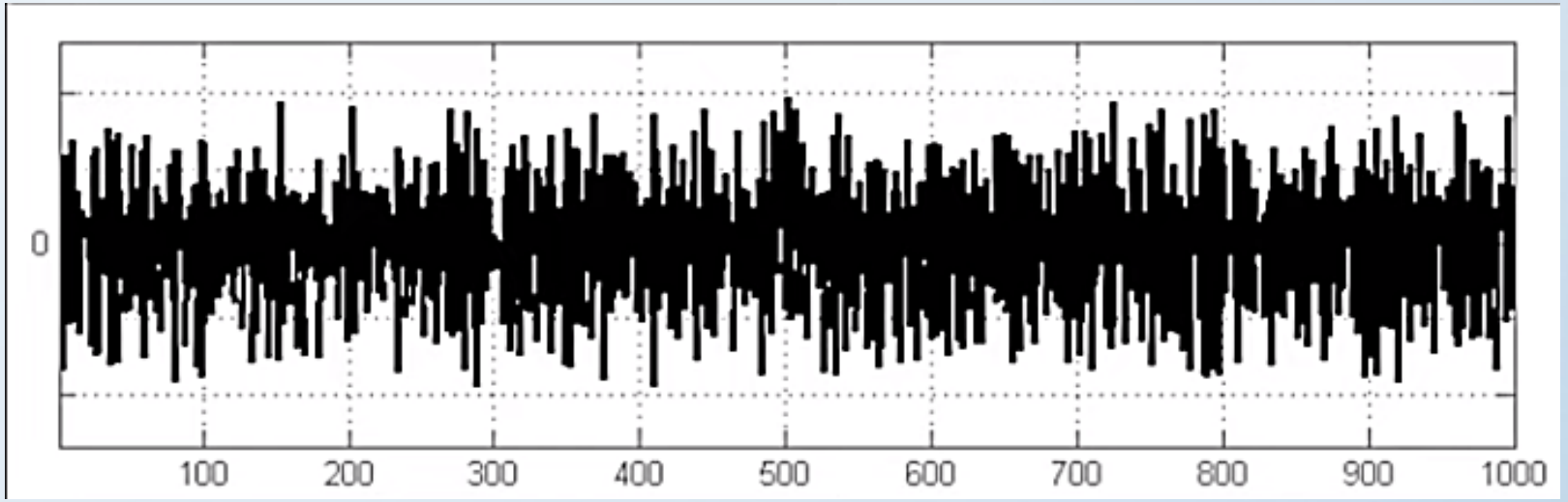
Laplacian
Zero-Crossing

Laplacian (∇^2) Edge Detector

$f(x)$



$\nabla f(x)$
(Gradient)



Where is the Edge ?

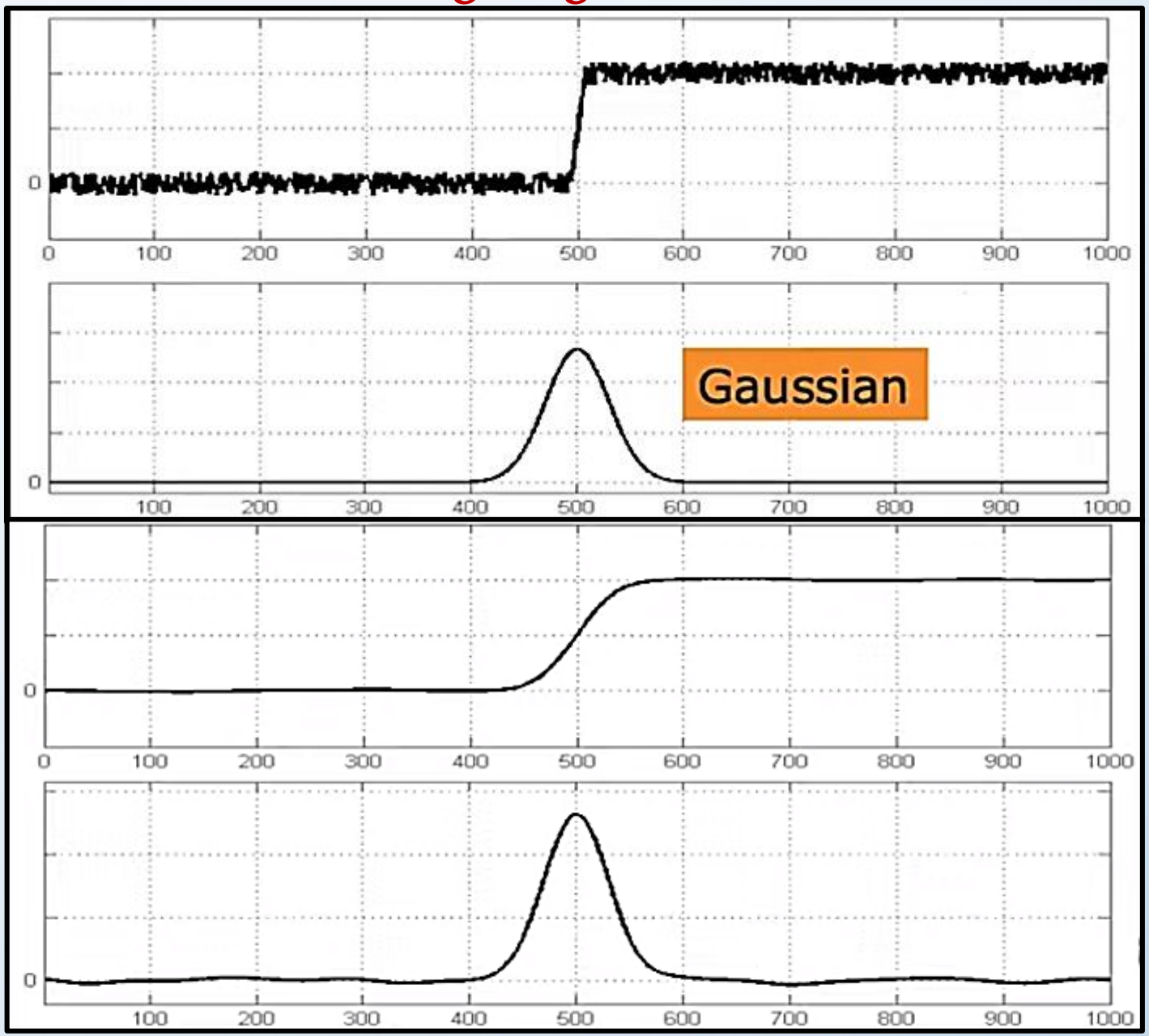
Low Pass Filtering using Gaussian Filter

f

η_σ
(Gaussian Filter)

$\eta_\sigma * f$
(Filtered Output)

$\nabla(\eta_\sigma * f)$
(Maxima as Edge)



Derivative of Gaussian (DoG) ($\nabla(\eta_\sigma * f)$)

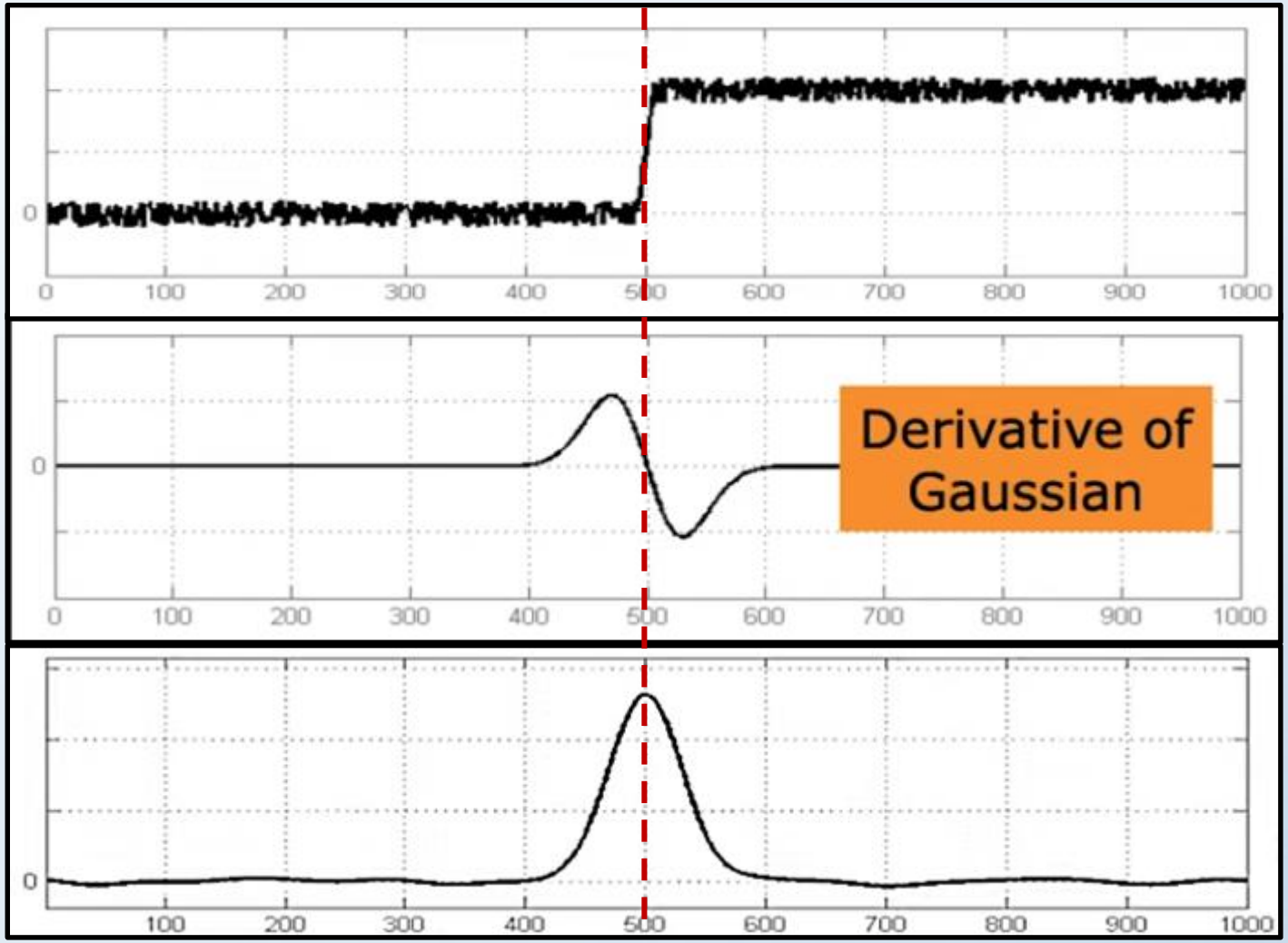
$\nabla(\eta_\sigma * f) = \nabla(\eta_\sigma) * f$: This operation save us one operation

Note: The derivative of Gaussian $\nabla(\eta_\sigma)$ is a constant and can be calculated before hand.

f

$\nabla(\eta_\sigma)$
(DoG Operator)

$\nabla(\eta_\sigma) * f$
(Maxima as Edge)



Laplacian of Gaussian (LoG) ($\nabla^2(\eta_\sigma * f)$)

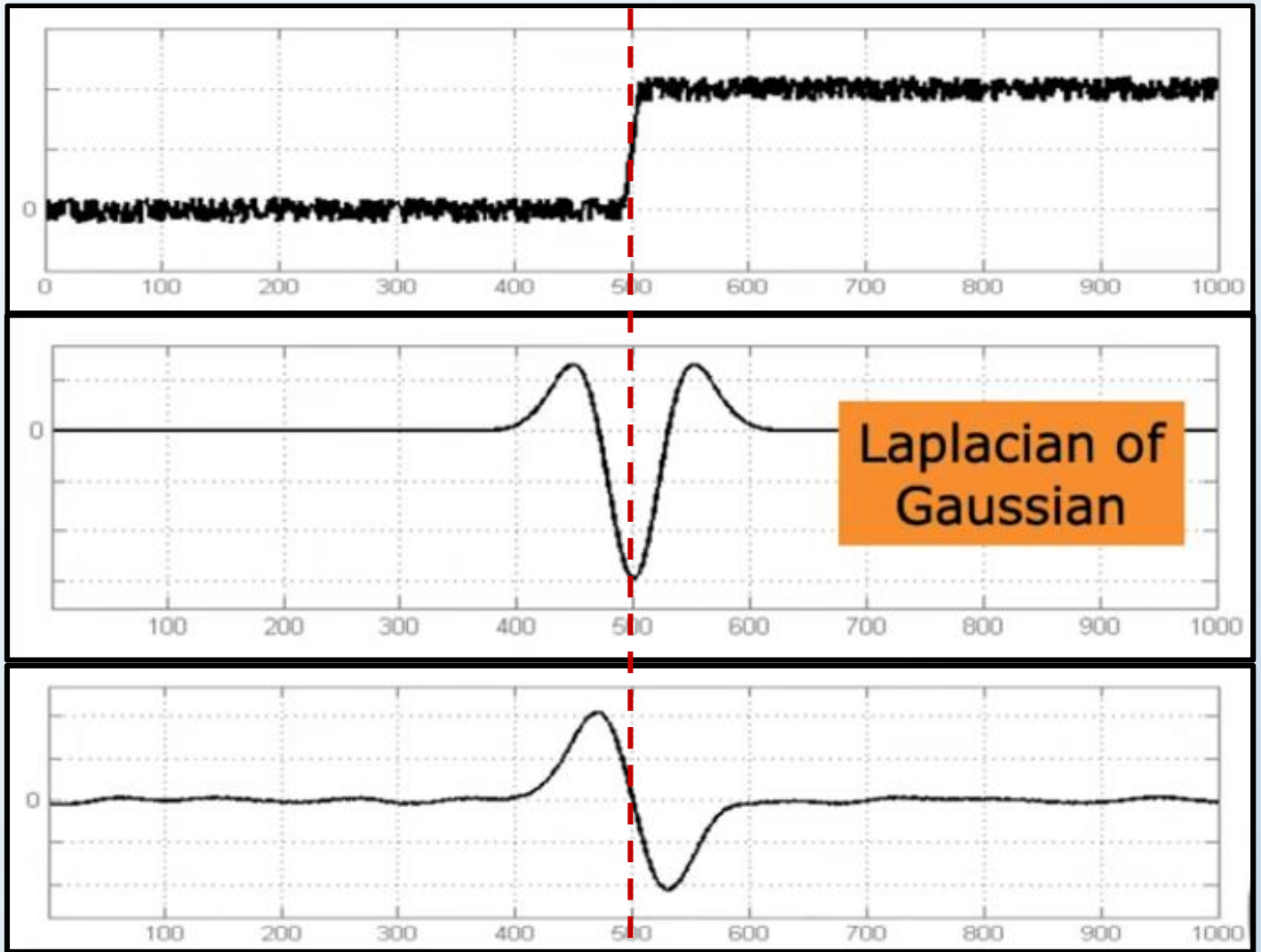
$\nabla^2(\eta_\sigma * f) = \nabla^2(\eta_\sigma) * f$: This operation save us one operation

Note: The derivative of Laplacian $\nabla^2(\eta_\sigma)$ is a constant and can be calculated before hand.

f

$\nabla^2(\eta_\sigma)$
(DoG Operator)

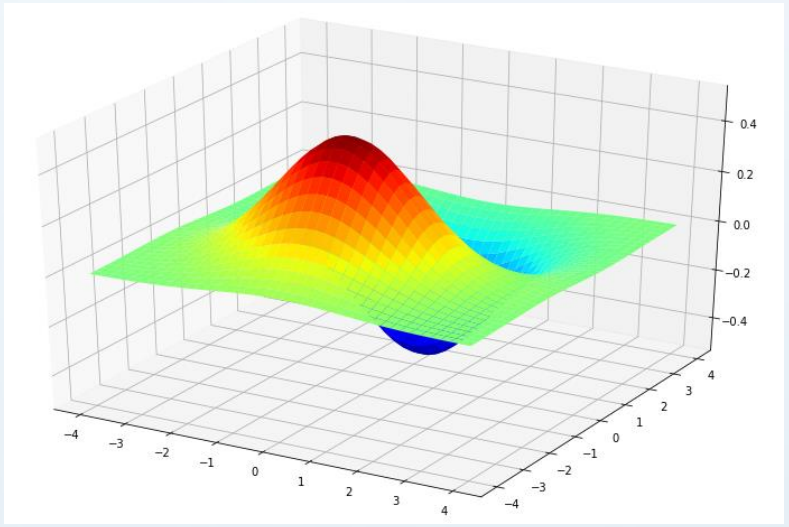
$\nabla^2(\eta_\sigma) * f$
(Zero-Crossing as Edge)



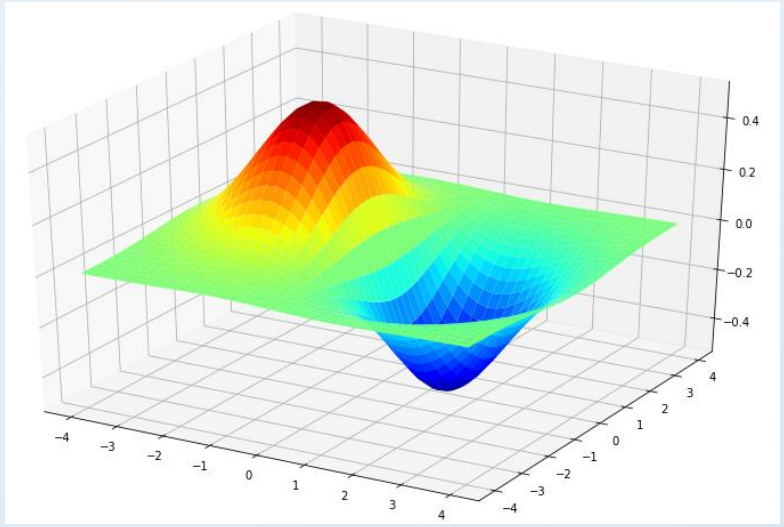
Gradient (DoG) vs. Laplacian (LoG)

Derivative of Gaussian

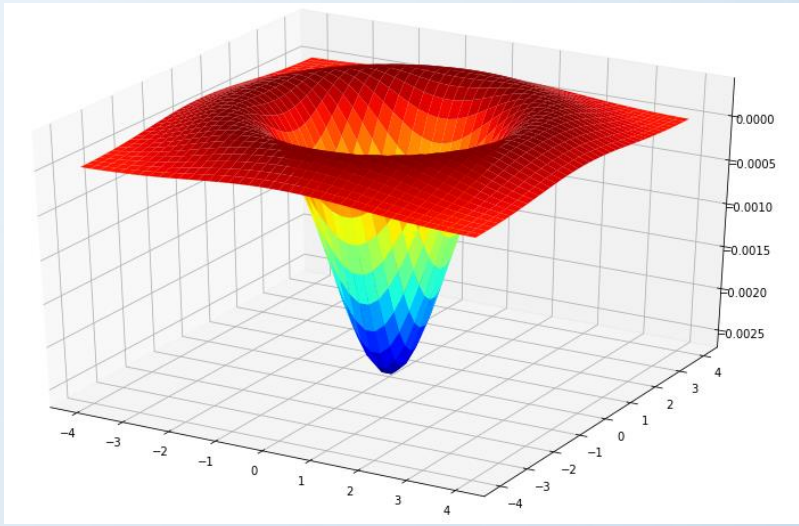
$$\frac{\partial}{\partial x}\,\eta_\sigma$$



$$\frac{\partial}{\partial y}\,\eta_\sigma$$



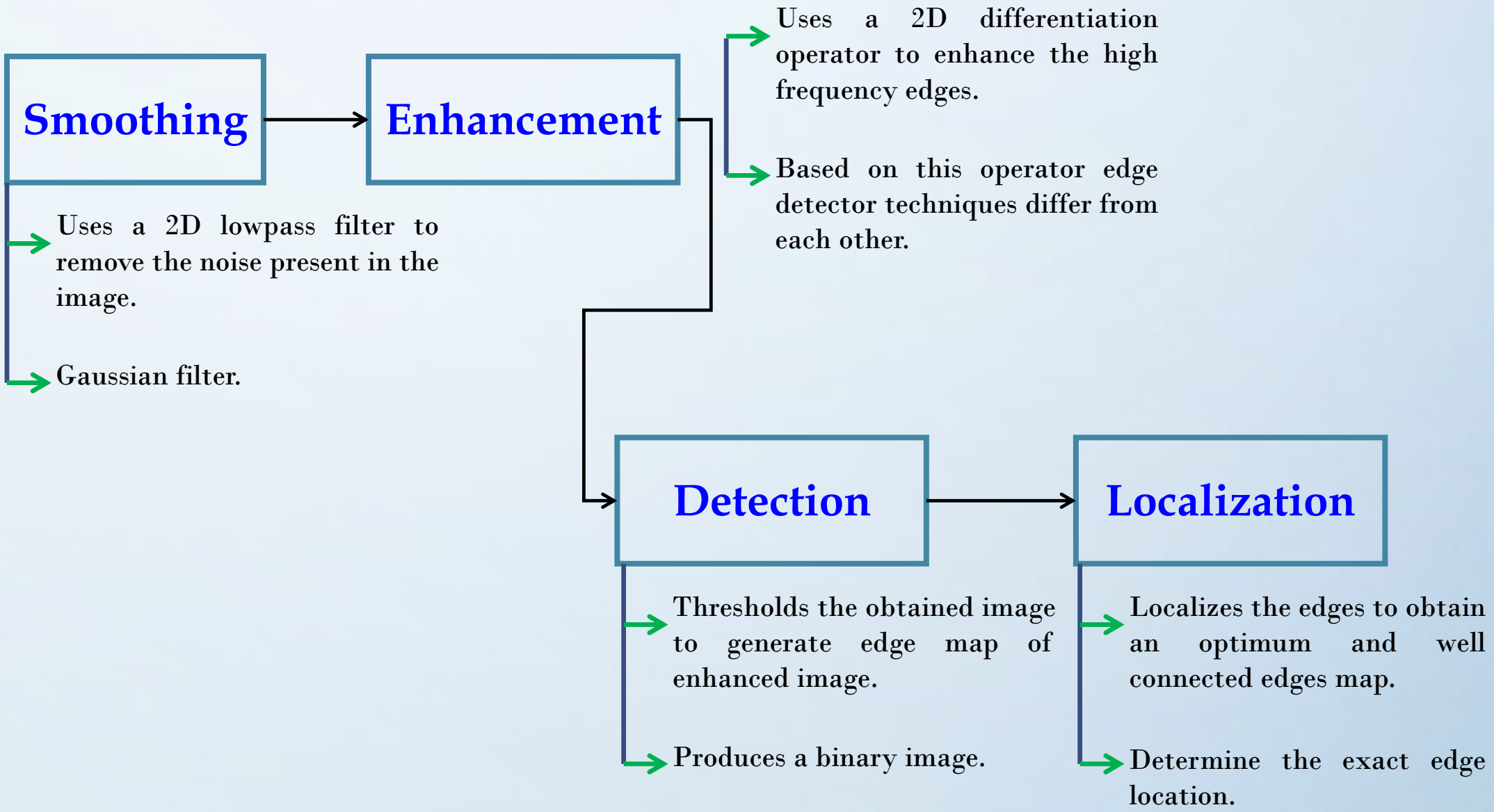
Laplacian of Gaussian



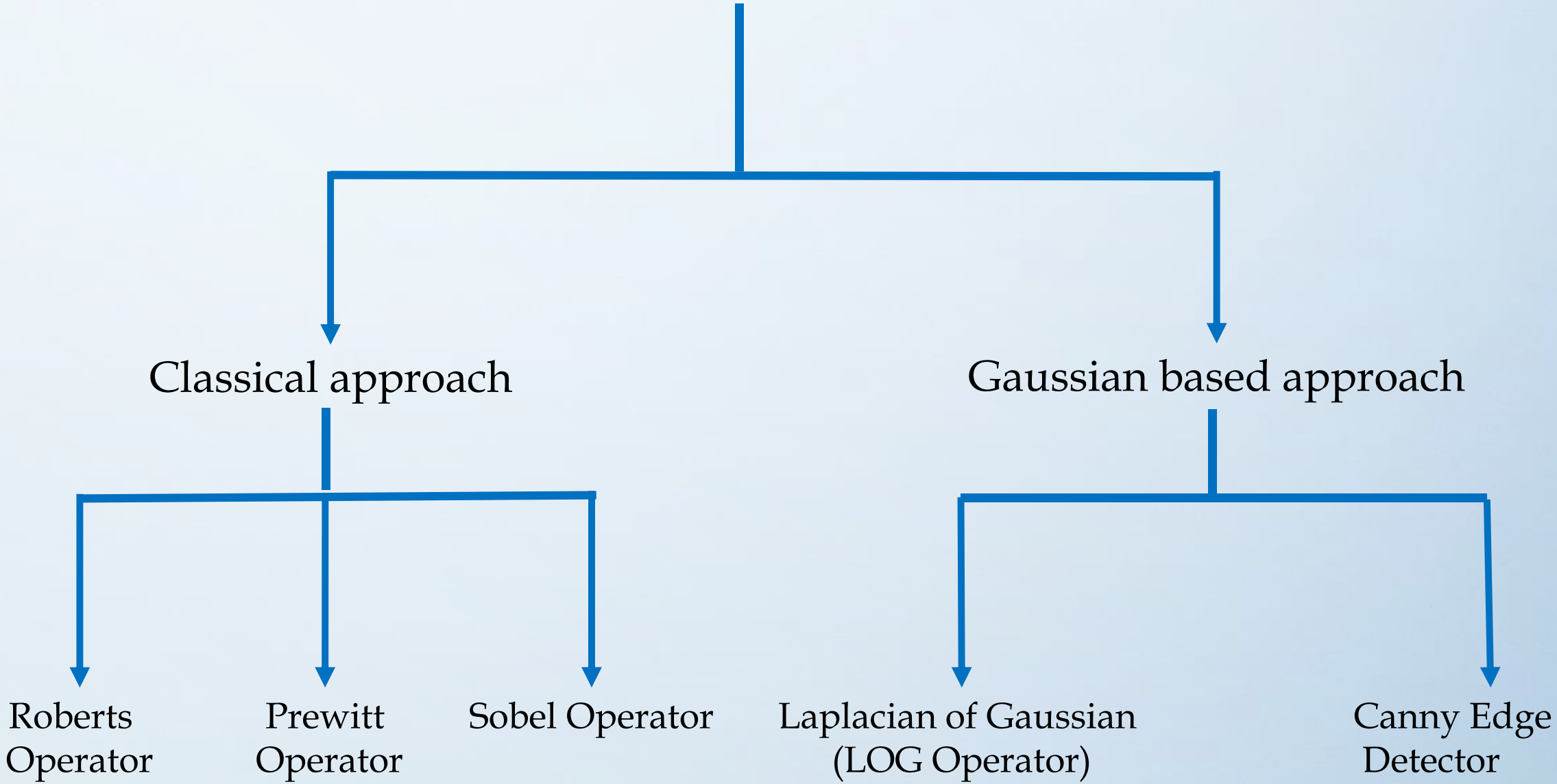
$$\frac{\partial^2}{\partial x^2}\,(\eta_\sigma) + \frac{\partial^2}{\partial y^2}\,(\eta_\sigma)$$

Gradient (DoG)	vs.	Laplacian (LoG)
Derivative of Gaussian		Laplacian of Gaussian
Provides location, magnitude and direction of edge.		Provides only location of edge.
Detection using maximum threshold.		Detection based on zero-crossing.
Non-linear operation. Requires two convolution.		Linear operation. Requires only one convolution.

Steps in Edge Detection



Approaches for Edge Detection



Sobel Operator

Image I

CONVOLUTION

$*$

-1	0	1
-2	0	2
-1	0	1

G_X

$\frac{d}{dx}I$

CONVOLUTION

$*$

-1	-2	-1
0	0	0
1	2	1

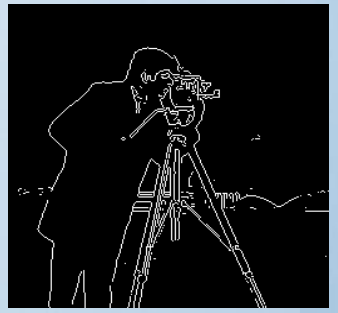
G_Y

$\frac{d}{dy}I$

$\sqrt{\left(\frac{d}{dx}I\right)^2 + \left(\frac{d}{dy}I\right)^2}$

Threshold

Edge Image

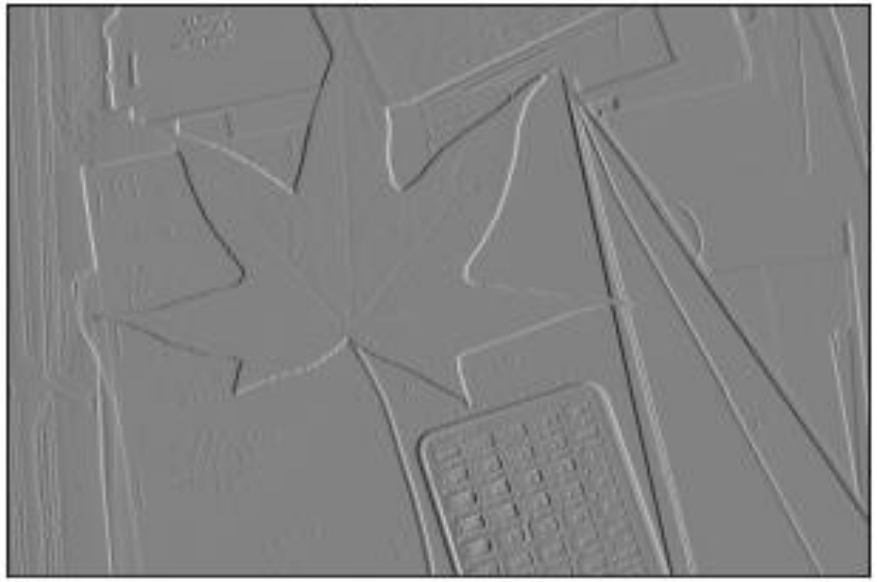


Sobel Operator

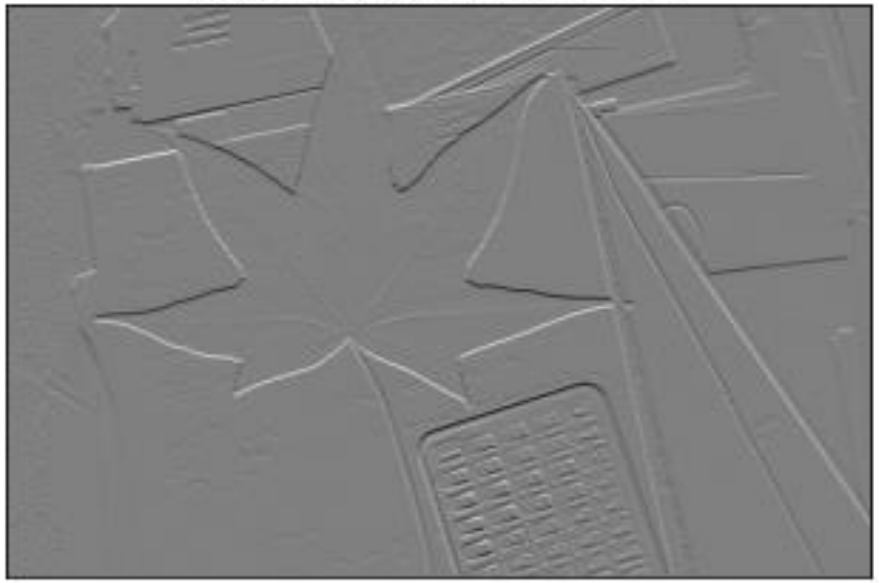
```
img = cv2.imread('/content/image_0040.jpg')
# Convert the image to grayscale image.
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

fig = plt.figure(figsize=(12,12))
fig.add_subplot(1,2,1)
plt.imshow(img_gray, cmap = 'gray')
plt.xticks([]), plt.yticks([])
plt.title('Input Image', fontweight = 'bold')
# Sobel Edge Detection on the X axis
img_SobEdgeX = cv2.Sobel(src=img_gray, ddepth=cv2.CV_64F, dx=1, dy=0, ksize=5)
fig.add_subplot(1,2,2)
plt.imshow(img_SobEdgeX, cmap = 'gray')
plt.xticks([]), plt.yticks([])
plt.title('Sobel Edge X-Dir No Filter', fontweight = 'bold')

fig = plt.figure(figsize=(12,12))
fig.add_subplot(1,2,1)
# Sobel Edge Detection on the Y axis
img_SobEdgeY = cv2.Sobel(src=img_gray, ddepth=cv2.CV_64F, dx=0, dy=1, ksize=5)
fig.add_subplot(1,2,1)
plt.imshow(img_SobEdgeY, cmap = 'gray')
plt.xticks([]), plt.yticks([])
plt.title('Sobel Edge Y-Dir No Filter', fontweight = 'bold')
# Sobel Edge Detection in the both direction
img_SobEdge = cv2.Sobel(src=img_gray, ddepth=cv2.CV_64F, dx=1, dy=1, ksize=5)
fig.add_subplot(1,2,2)
plt.imshow(img_SobEdge, cmap = 'gray')
plt.xticks([]), plt.yticks([])
plt.title('Sobel Edge Both-Dir No Filter', fontweight = 'bold')
```

Input Image

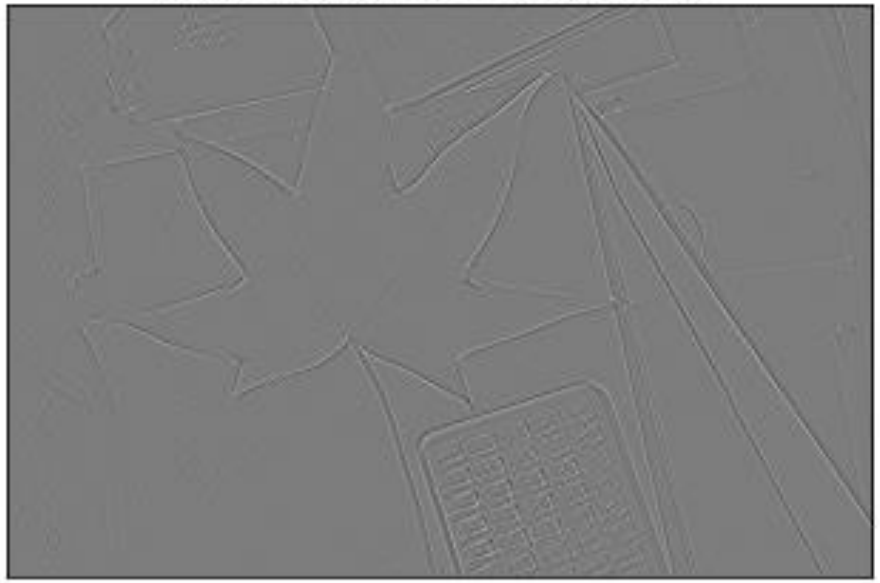
Sobel Edge X-Dir No Filter



Sobel Edge Y-Dir No Filter



Sobel Edge Both-Dir No Filter



Canny Edge Detector

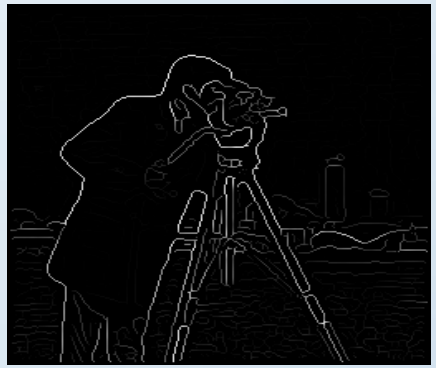
Image

Gaussian Filter

Horizontal gradient

Vertical gradient

Gradient Magnitude & Gradient Direction



Non maxima suppression

Hysteresis thresholding

Edge map

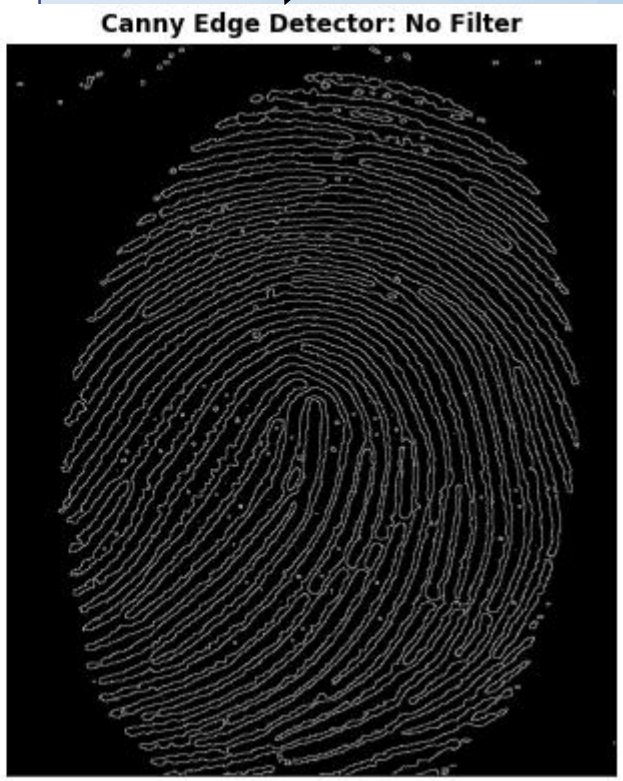
Canny Edge Detector

```
img_gray = cv2.imread('/content/Fig1038(a)(noisy_fingerprint).tif', 0)
# -------- CANNY EDGE on RAW IMAGE ---------------
EdgeCanny_Img = cv2.Canny(img_gray, 100, 200)
fig = plt.figure(figsize=(12,12))
fig.add_subplot(1,2,1)
plt.imshow(img_gray, cmap = 'gray')
plt.xticks([]), plt.yticks([])
plt.title('Input Image', fontweight = 'bold')

fig.add_subplot(1,2,2)
plt.imshow(EdgeCanny_Img , cmap = 'gray'
plt.xticks([]), plt.yticks([])
plt.title('Canny Edge Detector: No Filte
```

Edges Obtained From Canny Edge Detector; Thresh1 = 100 and Thresh2 = 200

Choose Various Lower and Upper Threshold for non-maximum Suppression

Input Image: No filtered applied



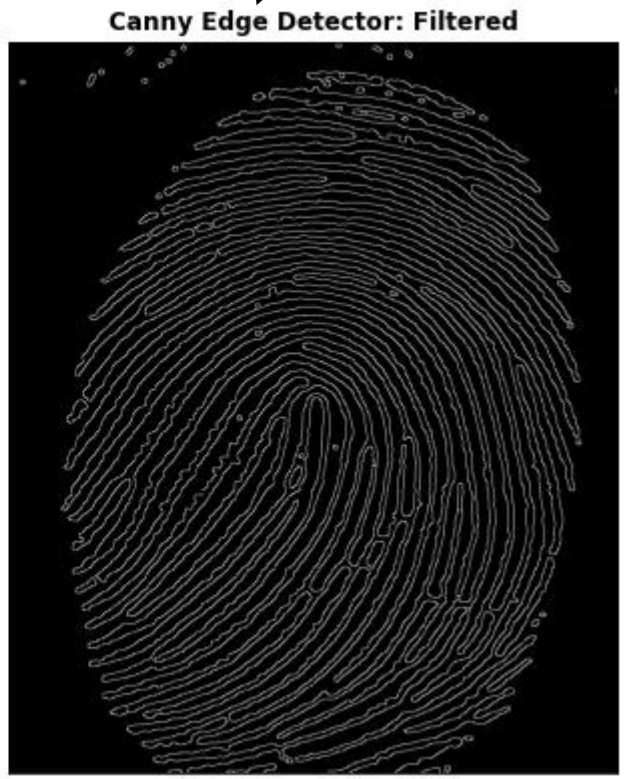
Canny Edge Detector

```
# ---- CANNY EDGE DETECTOR on FILTRED IMAGE ---------------
img_grayBlur = cv2.GaussianBlur(img_gray, (9, 9), 0)
EdgeCanny_ImgBlur = cv2.Canny(img_grayBlur, 100, 200)

fig = plt.figure(figsize=(12,12))
fig.add_subplot(1,2,1)
plt.imshow(img_grayBlur, cmap = 'gray')
plt.xticks([]), plt.yticks([])
plt.title('Input Image: Blurred', fontweight = 'bold')

fig.add_subplot(1,2,2)
plt.imshow(EdgeCanny_ImgBlur , cmap = 'gray
plt.xticks([]), plt.yticks([])
plt.title('Canny Edge Detector: Filtered',
```

Edges Obtained From Canny Edge Detector; Thresh1 = 100 and Thresh2 = 200

Gaussian Blur eliminated most of the unwanted edges detected at the center of the various lines of fingerprint

Input Image: Blurred Image



	Advantage	Disadvantage
Classical Operator	1) Simplicity 2) Computational complexity is very less	1) No smoothing operation so high sensitive to noise.
Gaussian Operator	1) Less sensitive to noise 2) Better edge detection	1) Complex Computation 2) Response time is high

Operator	Advantage	Disadvantage
LOG	Test wider areas around the pixels, Find the correct places of edges.	Fails at the corners and curves, where the intensity abruptly changes.
Canny	Improved signal to noise ratio, better edge detection in noise conditions	Complex computations, Response time is high.

Operator	Advantage	Disadvantage
Roberts	computational complexity is very less	Efficiency of edge detection is less.
Prewitt	Simplicity	The size of the kernel filter and coefficients are fixed
Sobel	Simplicity , the coefficients of masks are not fixed & can be adjusted acc. to our requirement .	Inaccurate, sensitive to noise

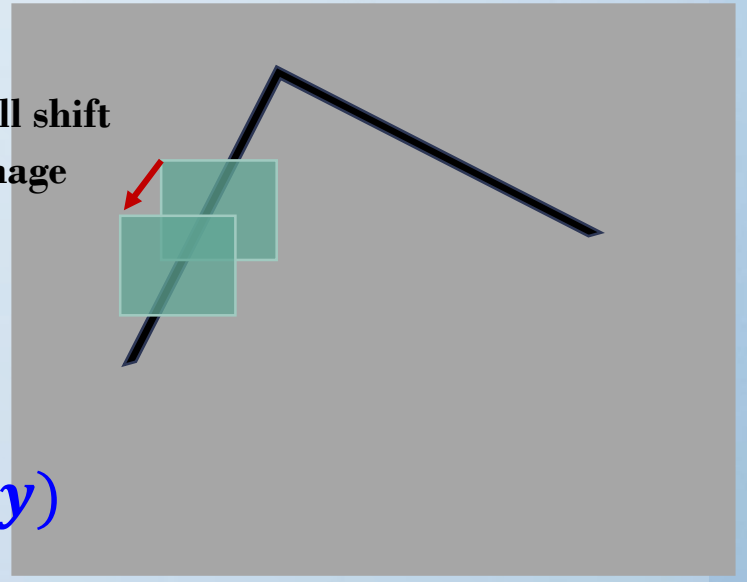
Corner Detector (Harris Corner Detector)

- First introduced by: Chris Harris and Mike Stephens in 1988.

- Basic Idea: Find locations such that the minimum change caused by shifting the window in any direction is large

Consider a small image patch $(x, y) \in w$(window) and shifting it by (u, v) in the image $I(x, y)$. AS we are suppose to find the direction of large change: we need to obtain the similarity (difference between) the patches as:

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



Harris Corner Detector (The math)

$I(x+u, y+v)$ can be approximated by Taylor's series: Assuming I_x and I_y as the partial derivative along x and y directions respectively:

$$I(x+u, y+v) \approx I(x,y) + I_x(x,y).u + I_y(x,y).v$$

- Pugging it in the previous equation and representing in matrix form:

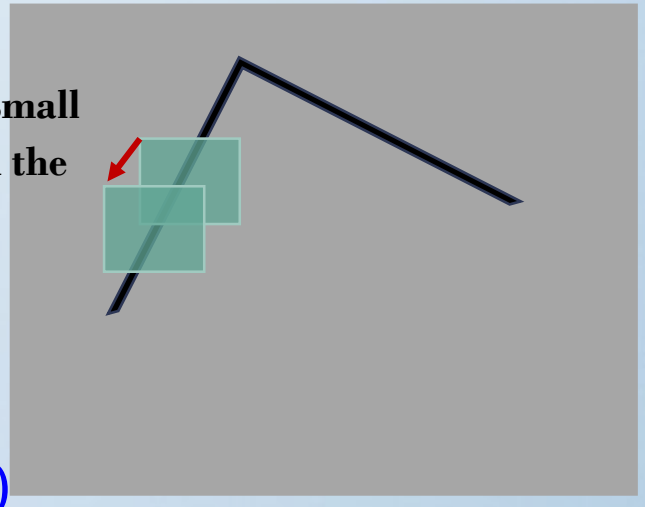
$$E(u,v) \approx \sum_{(x,y)\in W} \left[I(x,y) + \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} - I(x,y) \right]^2$$

$$\approx \sum_{(x,y)\in W} \left[\begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \right]^2$$

$$\approx \begin{bmatrix} u & v \end{bmatrix} \sum_{x,y\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

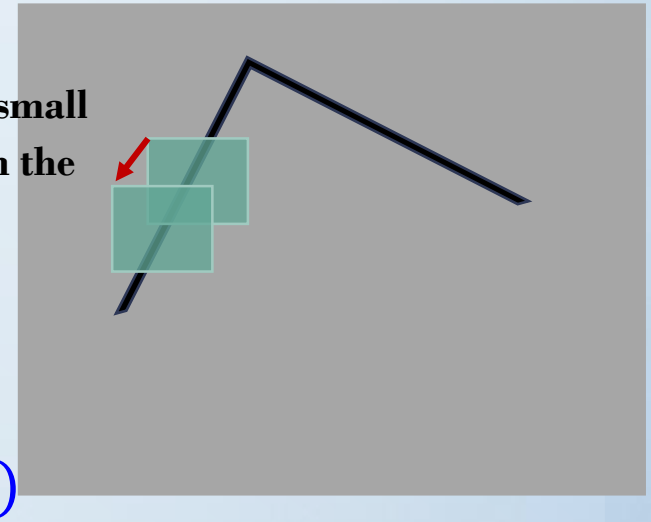
Structure Tensor

$$\boldsymbol{M}$$



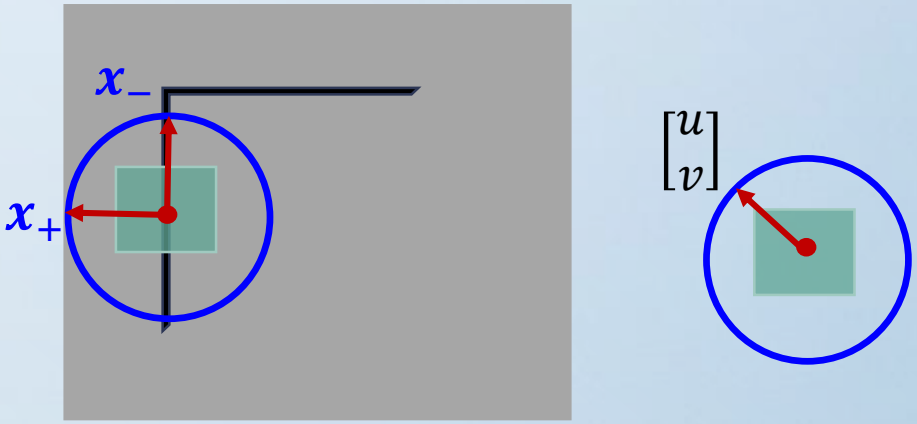
Harris Corner Detector (The math)

- The structure tensor M would suggest the direction of largest change.
- M represents the equation of ellipse.
- Two directions indicated by two Eigen values of M (as M is a 2×2 matrix.)



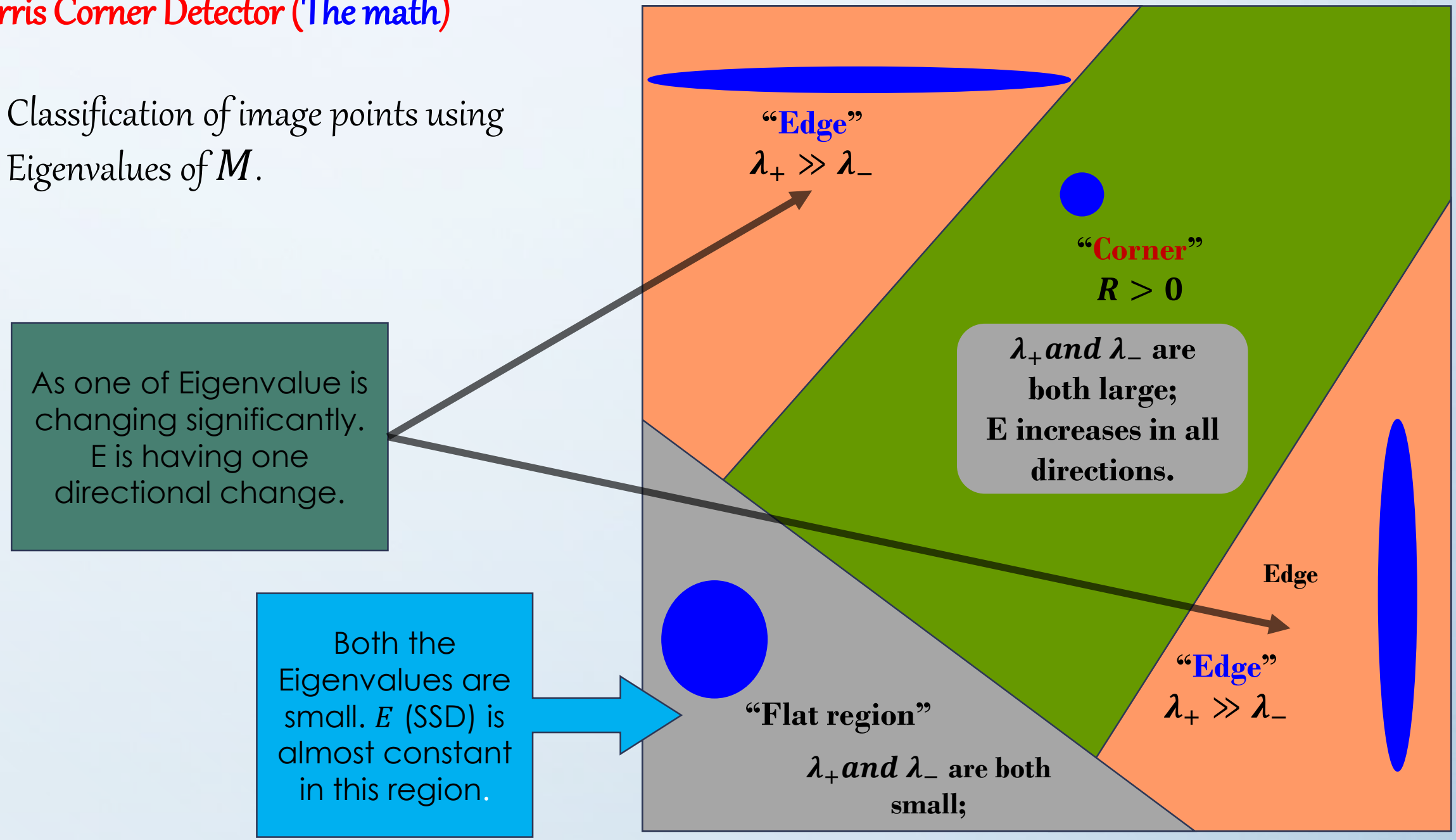
Harris Corner Detector (The math): Eigen Space of M

- Captures the smallest and largest changes due to window shift.
- x_+ = Direction of **largest** change in E.
- λ_+ = Magnitude of largest change in E.
- x_- = Direction of **Smallest** change in E.
- λ_- = Magnitude of Smallest change in E.



Harris Corner Detector (The math)

- Classification of image points using Eigenvalues of M.

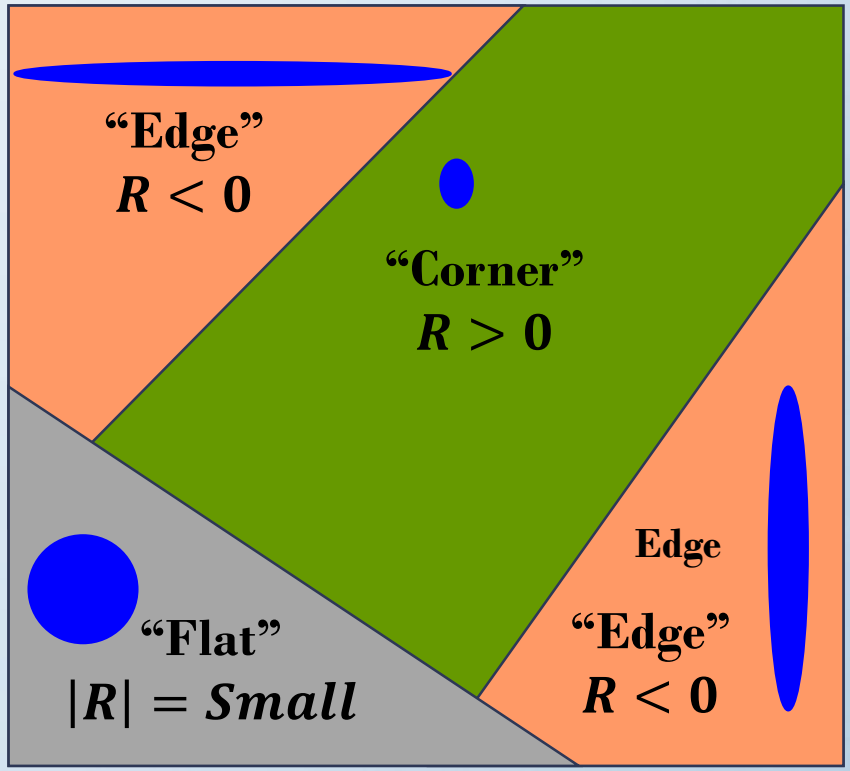


Harris Corner Detector (The math)

- Measure of cornerness or Harris response index can also be represented using λ_1 and λ_2 as:

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

Where: k is an empirical contact: $k \in [0.04, 0.06]$.



Harris Corner Detector (Python Implementation)

```
img = cv2.imread('/content/img1.bmp')
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
# Represent image in float32 format.
gray = np.float32(gray)
# Parameters of Harris Detector
# img - Input image. It should be grayscale and float32 type.
# blockSize - It is the size of neighborhood considered for corner detection
# ksize - Aperture parameter of the Sobel derivative used.
# k - Harris detector free parameter in the equation.
dst = cv2.cornerHarris(src=gray,blockSize=2,ksize=3,k=0.04)

#result is dilated for marking the corners, not important
dst = cv2.dilate(dst,None)

# Threshold for an optimal value, it may vary depending on
img[dst>0.05*dst.max()]=[0,0,255]
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
fig = plt.figure(figsize = (12,12))
plt.imshow(img)
plt.title('Detected Corners using Harris Operator', fontwe
```

Pic Courtesy: The Cityscapes Dataset: https://www.cityscapes-dataset.com/