Feature Detection and Matching

Objectives

To learn and appreciate the following concepts:

- ✓Introduction to Edge detection methods (Laplacian & Canny edge)
- ✓Points and Patches, Harries corner detection methods
- ✓Gradian & Gaussian Detector Techniques
- ✓SIFT, COLOR & Texture, RANSAC models

Session outcome

- At the end of session the student will be able to understand:
  - What is Edge detection methods
  - Fundamentals of Features like - point-based, color-based, Texture-based methods
  - Importance of Harris corner detection methods
  - Understanding SIFT, RANSAC methods

- SIFT – Scale-Invariant Feature Transform
- RANSAC - Random Sample Consensus

Applications of Feature Detection and Matching

- Automatic Object Tracking
- Point matching for computing disparity
- Stereo calibration(Estimation of the fundamental matrix)
- Motion-based segmentation
- Recognition
- 3D object reconstruction
- Robot navigation
- Image retrieval and indexing

Feature Meaning

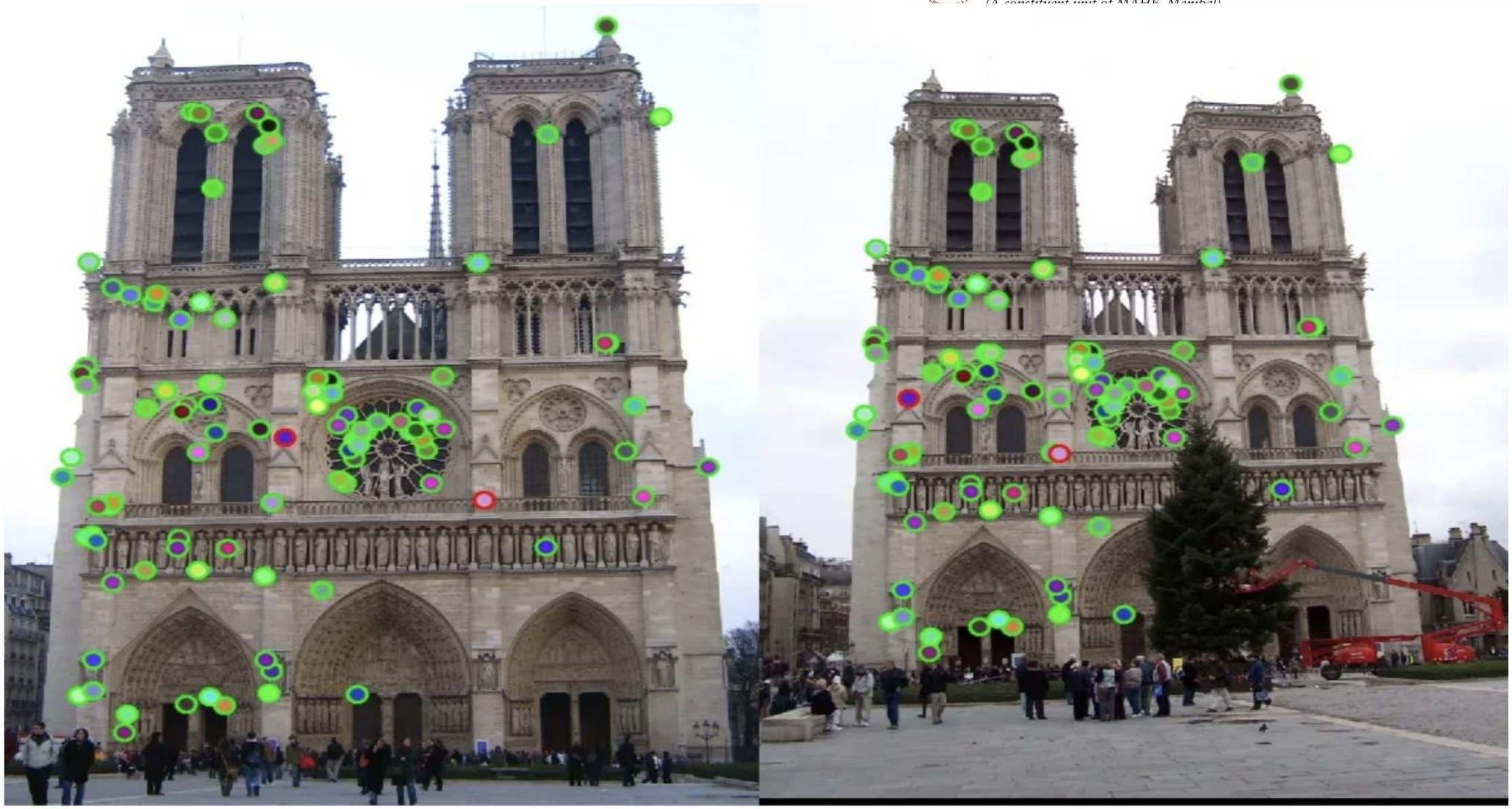
- A feature is a piece of information which is relevant for solving the computational task related to a certain application.
- Features may be specific structures in the image such as points, edges or objects.
- Features may also be the result of a general neighborhood operation or feature detection applied to the image.
- The features can be classified into two main categories:
  - The features that are in specific locations of the images, such as mountain peaks, building corners, doorways, or interestingly shaped patches of snow. These kinds of localized features are often called **keypoint features** (or even corners) and are often described by the appearance of patches of pixels surrounding the point location.
  - The features that can be matched based on their orientation and local appearance (edge profiles) are called **edges** and they can also be good indicators of object boundaries and occlusion events in the image sequence.

Main Component of Feature Detection and Matching

- **Detection:** Identify the **Interest Point**
- **Description:** The local appearance around each feature point is described in some way that is (ideally) invariant under changes in illumination, translation, scale, and in-plane rotation. We typically end up with a descriptor vector for each feature point.
- **Matching:** Descriptors are compared across the images, to identify similar features. For two images we may get a set of pairs:

$$(\boldsymbol{Xi, Yi}) \leftrightarrow (\boldsymbol{Xi`, Yi`}),$$

where (***Xi, Yi***) is a feature in one image and (***Xi`, Yi`***) its matching feature in the other image.



What is Edge Detection ?

- Edge detection is a technique of image processing used to identify points in a digital image with discontinuities, simply to say, sharp changes in the image brightness.

- These points where the image brightness varies sharply are called the edges (or boundaries) of the image.

- Edge detection is mostly used for the measurement, detection and location changes in an image gray. Edges are the basic feature of an image. In an object, the clearest part is the edges and lines.

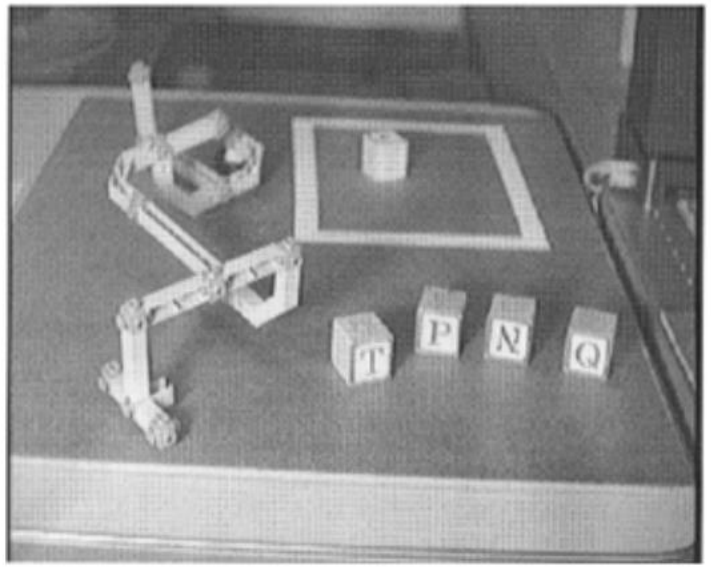
Edge Detection

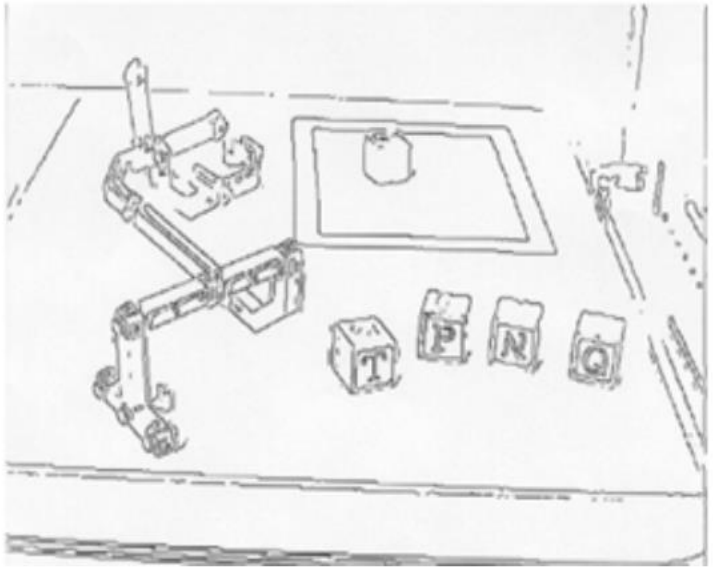
- The basic idea behind edge detection is as follows:

➢To highlight local edge operator use edge enhancement operator.

➢Define the edge strength and set the edge points.

NOTE: edge detection cannot be performed when there are noise and blurring image.

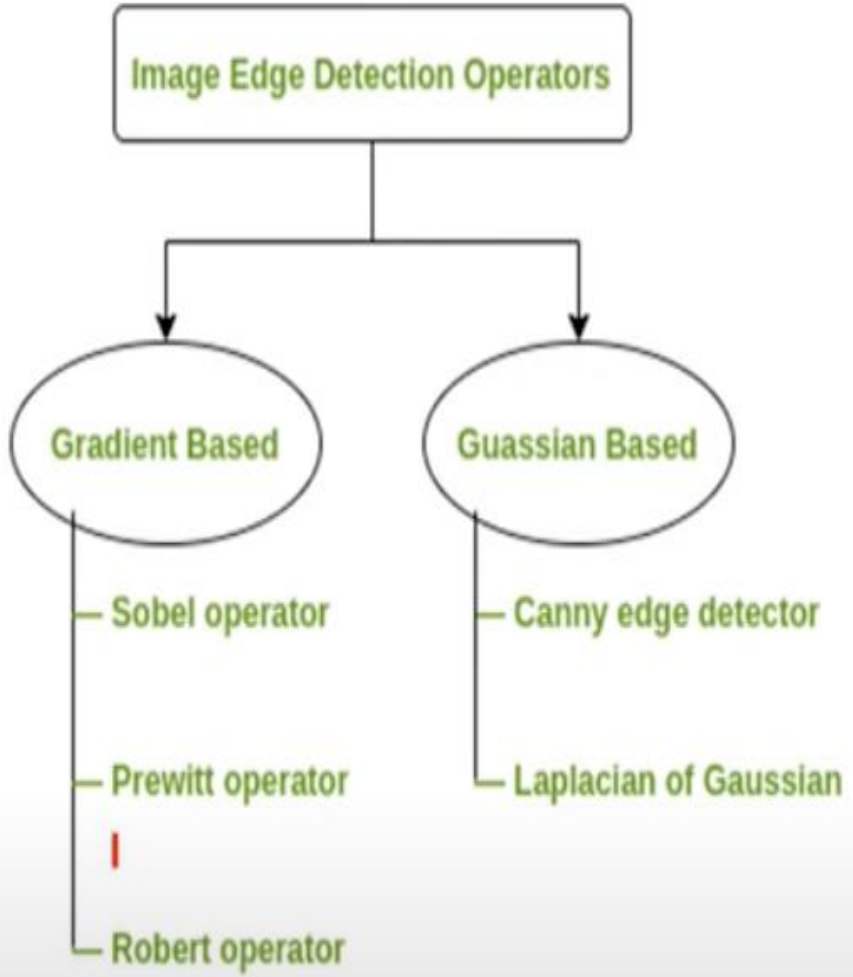




Edge Detection Methods

Gradient – based operator which computes first-order derivations in a digital image like

- Sobel operator,
- Prewitt operator,
- Robert operator



(a) Sobel Edge Detection

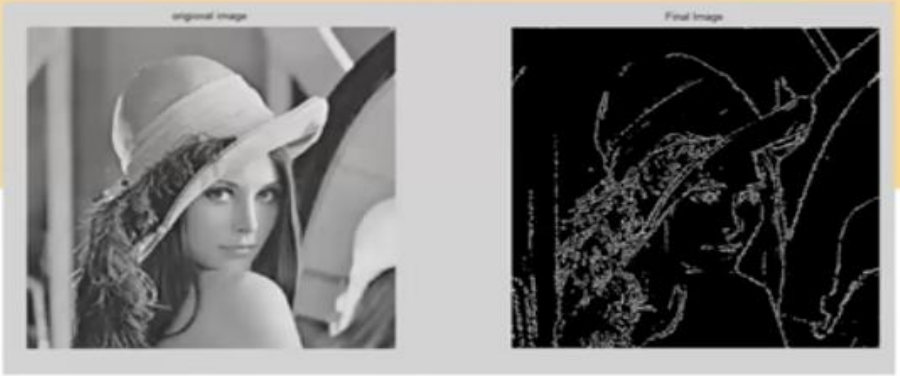
- The Sobel edge detection operator extracts all the edges of an image, without worrying about the directions. The main advantage of the Sobel operator is that it provides differencing and smoothing effect.

- Sobel edge detection operator is implemented as the sum of two directional edges. And the resulting image is a unidirectional outline in the original image.

(a) Sobel Edge Detection

Sobel Operator:

- It uses two 3 x 3 kernels or masks which are convolved with the input image to calculate the vertical and horizontal derivative approximations respectively



Vertical direction

1	0	1
-2	0	2
-1	0	1

G_x

We can apply more weight to mask

-1	0	1
-5	0	5
-1	0	1

Horizontal Direction

1	-2	-1
0	0	0
1	2	1

G_y

$$|G| = \sqrt{Gx^2 + Gy^2}$$

(a) Sobel Edge Detection

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

Gx

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

Gy

Sobel Edge detection operator consists of 3x3 convolution kernels. Gx is a simple kernel and Gy is rotated by 90°

These Kernels are applied separately to input image because separate measurements can be produced in each orientation i.e Gx and Gy.

Following is the gradient magnitude:

$$|G| = \sqrt{Gx^2 + Gy^2}$$

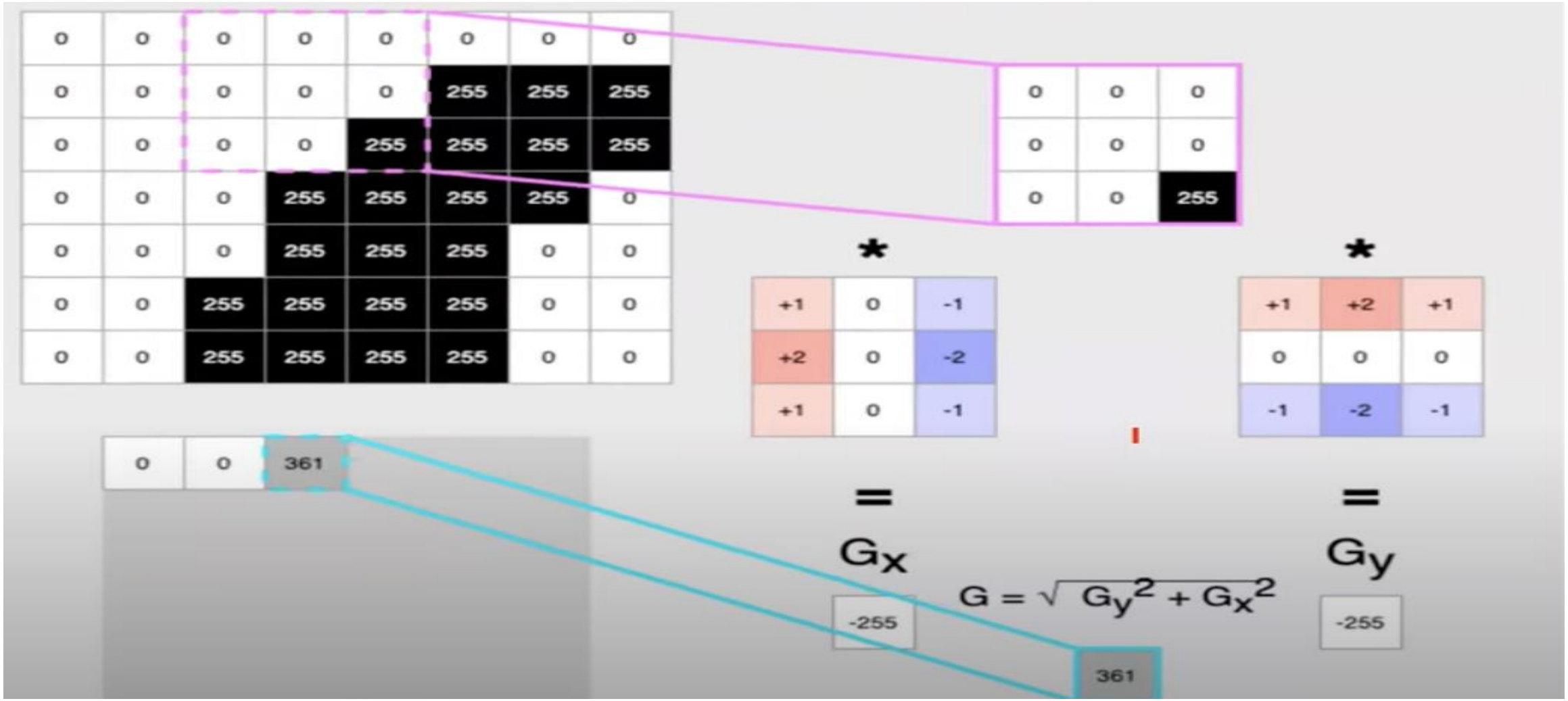
As it is much faster to compute An approximate magnitude is computed:

$$|G| = |Gx| + |Gy|$$

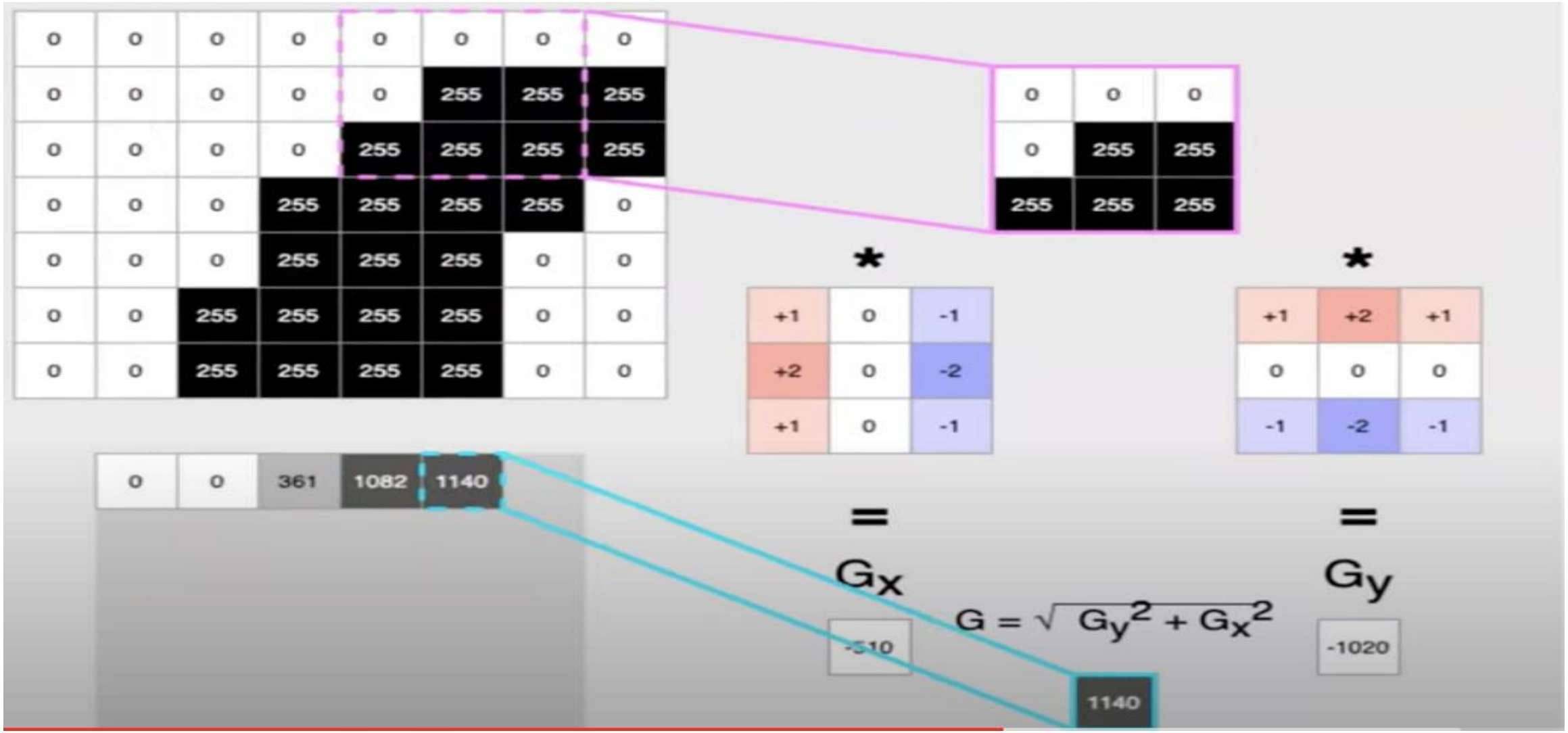
The angle of orientation of the edge (relative to the pixel grid) giving rise to the spatial gradient is given by:

$$\theta = \arctan(Gy/Gx)$$

Method of calculation



Method of calculation

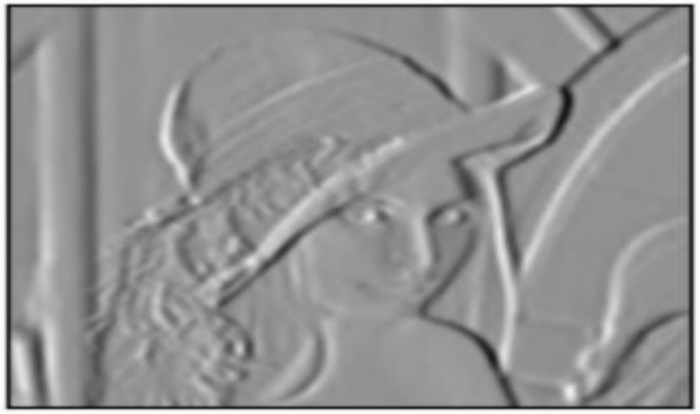


(a) Sobel Edge Detection

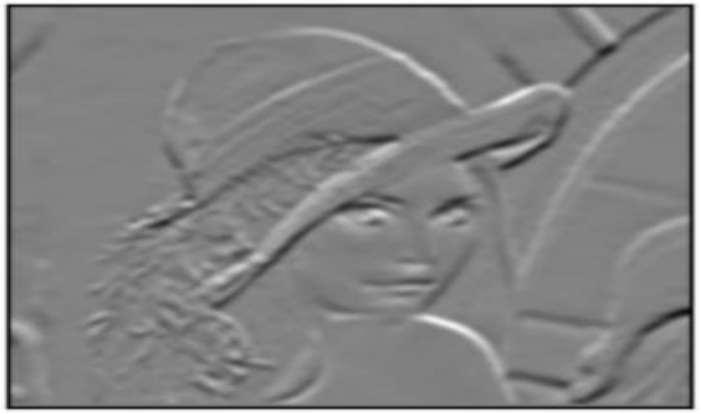
Original Image

Sobel X Y using Sobel() function

sobelx



sobely



(b) Prewitt Operator

Prewitt Operator:

- This operator is almost similar to the sobel operator.
- It also detects vertical and horizontal edges of an image.
- Prewitt operator provides us **two masks** one for detecting edges in horizontal direction and another for detecting edges in an vertical direction.

Vertical direction

-1	0	1
-1	0	1
-1	0	1

G_x

Horizontal Direction

-1	-1	-1
0	0	0
1	1	1

G_y

$$|G| = \sqrt{Gx^2 + Gy^2}$$

(a) Original

(d) Sobel

(e) Prewitt

(c) Robert Cross Edge Detector

The operator consists of a pair of 2×2 convolution kernels

1	0
0	-1

G_x

0	1
-1	0

G_y

$$|G| = \sqrt{Gx^2 + Gy^2}$$

(a) Original

(c) Robert

(d) Sobel (e) Prewitt

Line Detection

- A special mask is needed to detect a special type of line
- Examples:
  - Horizontal mask has high response when a line passed through the middle row of the mask.

−1	−1	−1
2	2	2
−1	−1	−1

Horizontal

−1	−1	2
−1	2	−1
2	−1	−1

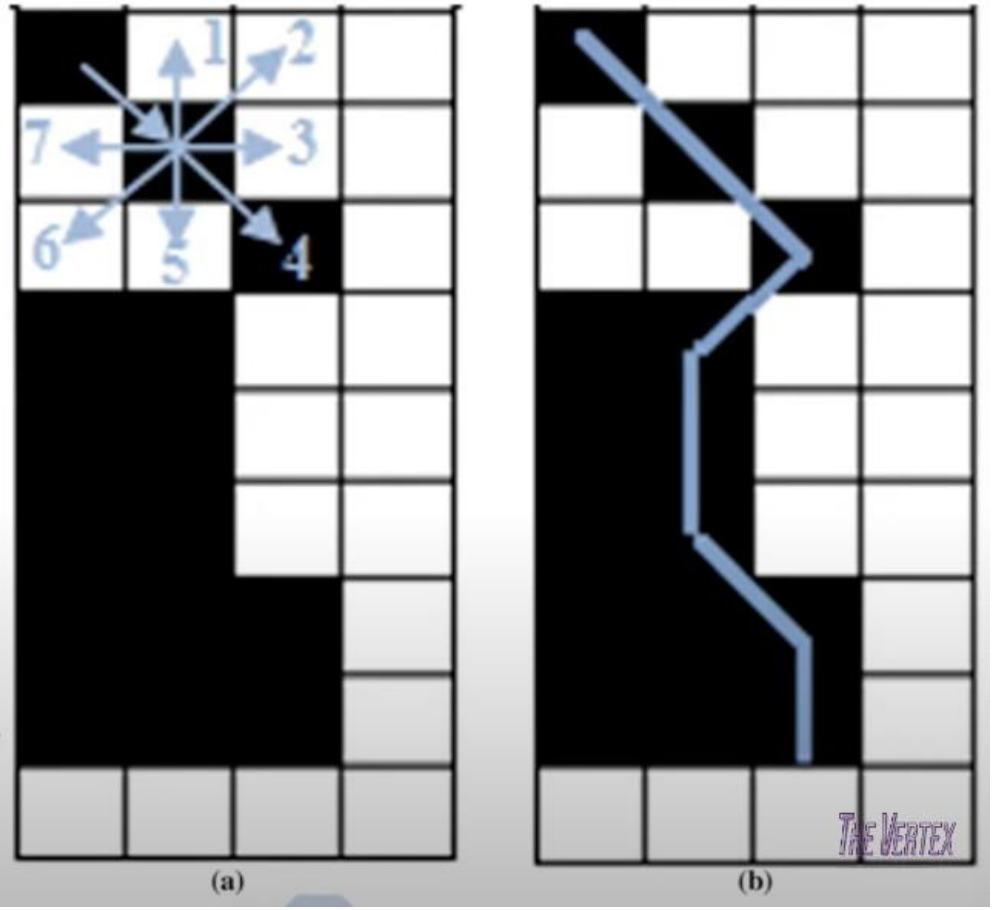
+45°

−1	2	−1
−1	2	−1
−1	2	−1

Vertical

2	−1	−1
−1	2	−1
−1	−1	2

−45°



(d) Laplacian Detection

- The Laplacian of Gaussian is a 2-D isotropic measure of an image.
- In an image, Laplacian is the highlighted region in which rapid intensity changes and it is also used for edge detection.
- The Laplacian is applied to an image which is been smoothed using a Gaussian smoothing filter to reduce the sensitivity of noise.
- This operator takes a single grey level image as input and produces a single grey level image as output.

(d) Laplacian Detection

In Laplacian, the input image is represented as a set of discrete pixels. So discrete convolution kernel which can approximate second derivatives in the definition is found.

3 commonly used kernels are as following:

0	1	0
1	-4	1
0	1	0

1	1	1
1	-8	1
1	1	1

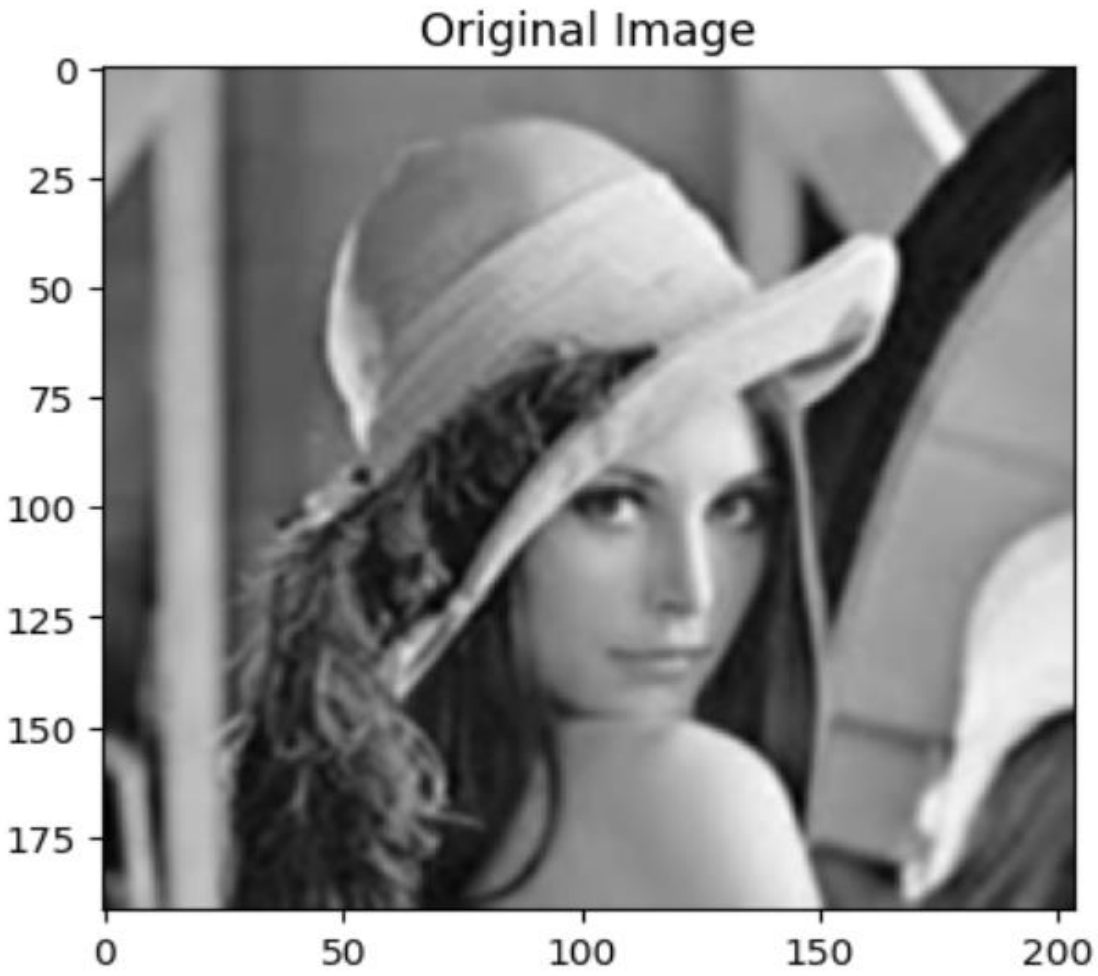
-1	2	-1
2	-4	2
-1	2	-1

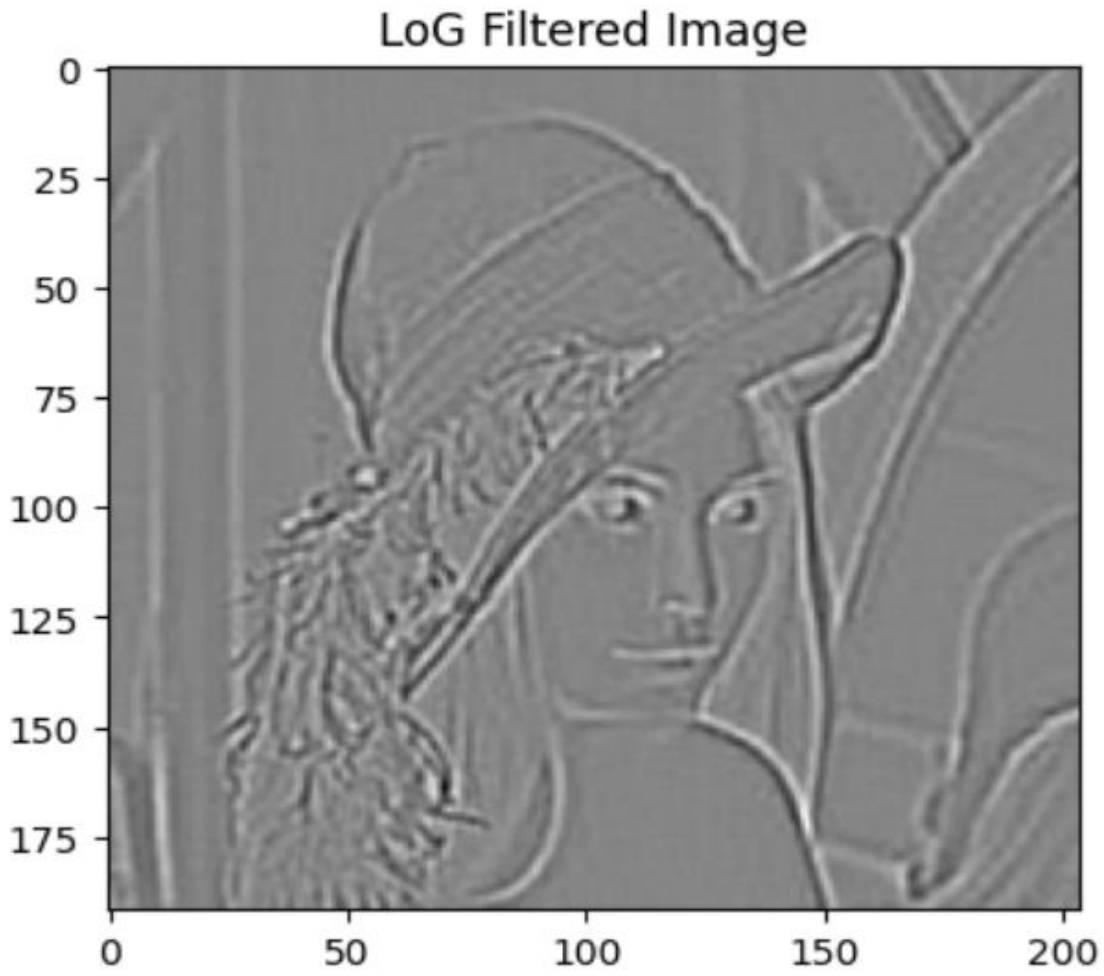
This is 3 discrete approximations which are used commonly in Laplacian filter.

Following is 2-D Log function with Gaussian standard deviation:

$$LoG(x, y) = -\frac{1}{\Pi\sigma^2}\left[1 - \frac{x^2 + y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

(d) Laplacian Detection


Original Image
0
25
50
75
100
125
150
175
0
50
100
150
200


LoG Filtered Image
0
25
50
75
100
125
150
175
0
50
100
150
200

(e) Canny Edge Detection

- Canny Edge Detection is a popular edge detection algorithm. It was developed by John F. Canny.
- It is a multi-stage algorithm.

➢ **Noise Reduction**

- Since edge detection is susceptible to noise in the image, first step is to remove the noise in the image with a 5x5 Gaussian filter.

➢ **Finding Intensity Gradient of the Image**

- Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction (Gx) and vertical direction (Gy). From these two images, we can find edge gradient and direction for each pixel as follows:

$$\text{Edge\_Gradient (G)} = \sqrt{G_x^2 + G_y^2}$$

$$\text{Angle } (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

- Gradient direction is always perpendicular to edges. It is rounded to one of four angles representing vertical, horizontal and two diagonal directions.

(e) Canny Edge Detection

Original Image

Canny Edge Detection Image

Image Features

- Feature is the piece of information. Features may be certain structures in an image like colors, points, edges, corners, or objects.
- Two classes: (a) Keypoint features (b) Edges

(a) Keypoint Features:

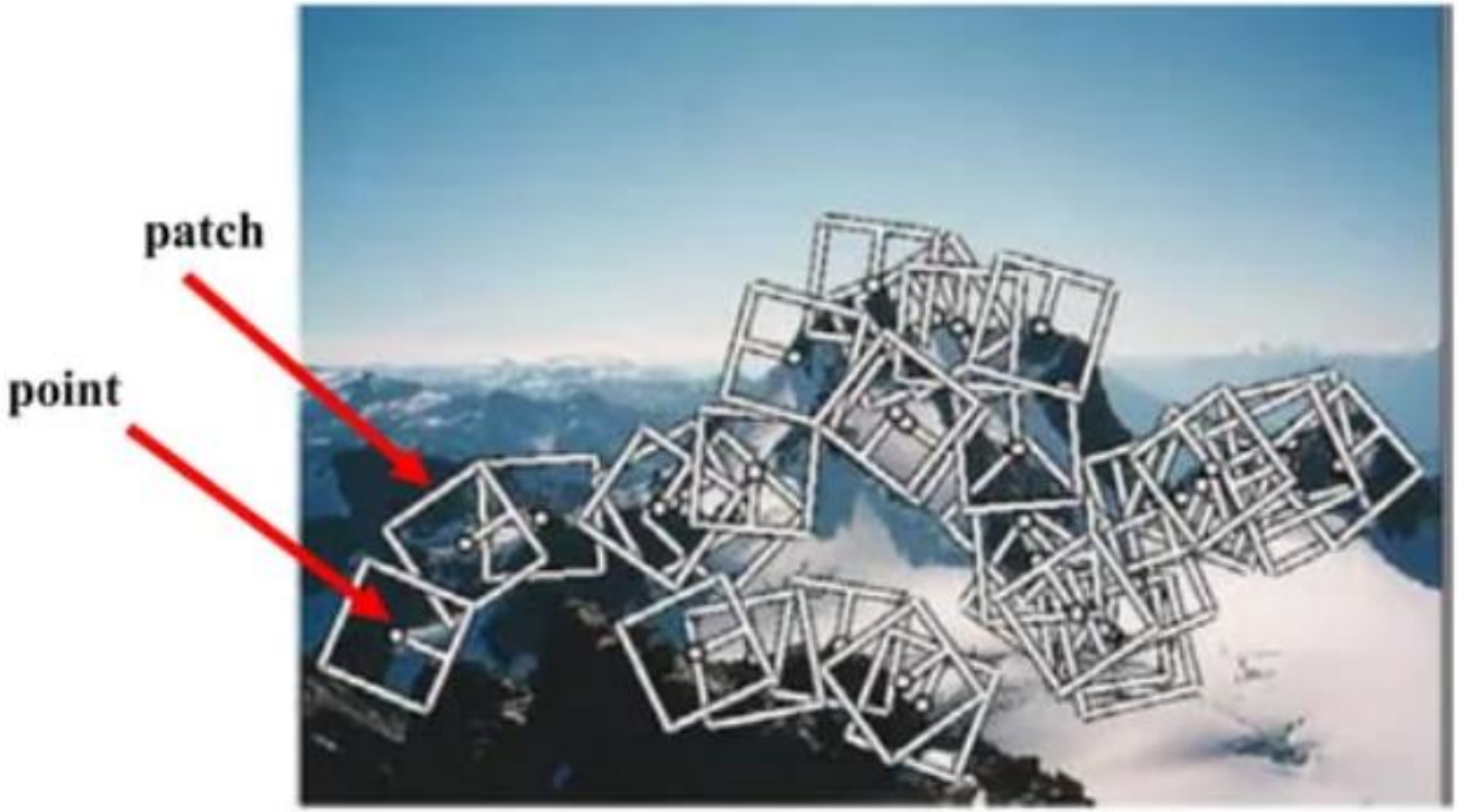
- Keypoint features are specific locations in the images such as mountains, peaks, building corners, doorways, or interesting shaped patches.
- These localized features are called keypoint features or interest points or corners.
- The keypoint features are described by the appearance of patches of pixels surroundings the point location.

Image Features

(b) Edges:

- These features can be matched based on their orientation and local appearance (Edge Profiles).
- Edge features are good indicators of object boundaries and occlusion (blockage) events in image sequences.
- Edges can be grouped into longer curves and straight line segments, which are be directly matched or analyzed to find vanishing points, internal and external camera parameters.
- Point, edges and lines features provide information to both keypoint and region-based descriptors to describe object boundaries and man-made objects.

Image Features



Components of Feature Detection and Matching

➢ **Feature Detection (Extraction):**

- Each image is searched for locations (interest point) that are likely to match well in other images.

➢ **Feature Description:**

- Each region around detected keypoint locations is converted into a more compact and stable (invariant) descriptor under changes in illumination, scale, translation, rotation that can be matched against other descriptors.

➢ **Feature Matching:**

- Efficient searches of likely matching candidates (feature descriptors) in other images.

Invariant Local Features

- Features invariant to transformations are:

(a) Geometric Invariance: Translation, Rotation, Scaling.

(b) Photometric Invariance: Brightness, Exposure etc.

Advantages of Local Features:

- **Locality:**
  - Features are local, so robust to occlusion and cluster.
- **Distinctiveness:**
  - Can differentiate a large database of objects.
- **Quantity:**
  - Hundreds or thousands in a single image.
- **Efficiency:**
  - Real-time performance achievable.
- **Generality:**
  - Exploit different types of features in different situations.

Point Features/Corners

- Interest point is the point at which the direction of the boundary of the object changes abruptly or intersection point between two or more edge segments.
- Point features are used to find a sparse set of corresponding locations in **different images** as a pre-cursor to computing camera pose.
- It is a pre-requisite for computing a denser set of correspondences using stereo matching.
- Denser set of correspondences are used to **align** different images, e.g., when stitching image mosaics or performing video stabilization.
- They are also used extensively to perform object instance and category recognition.
- Advantage of key-points is matching the images even in the presence of cluster(occlusion), large-scale and orientation changes.
- Feature-based correspondence techniques used in **stereo-matching**, **image-stitching, automated 3Dmodelling** applications.

Point Features

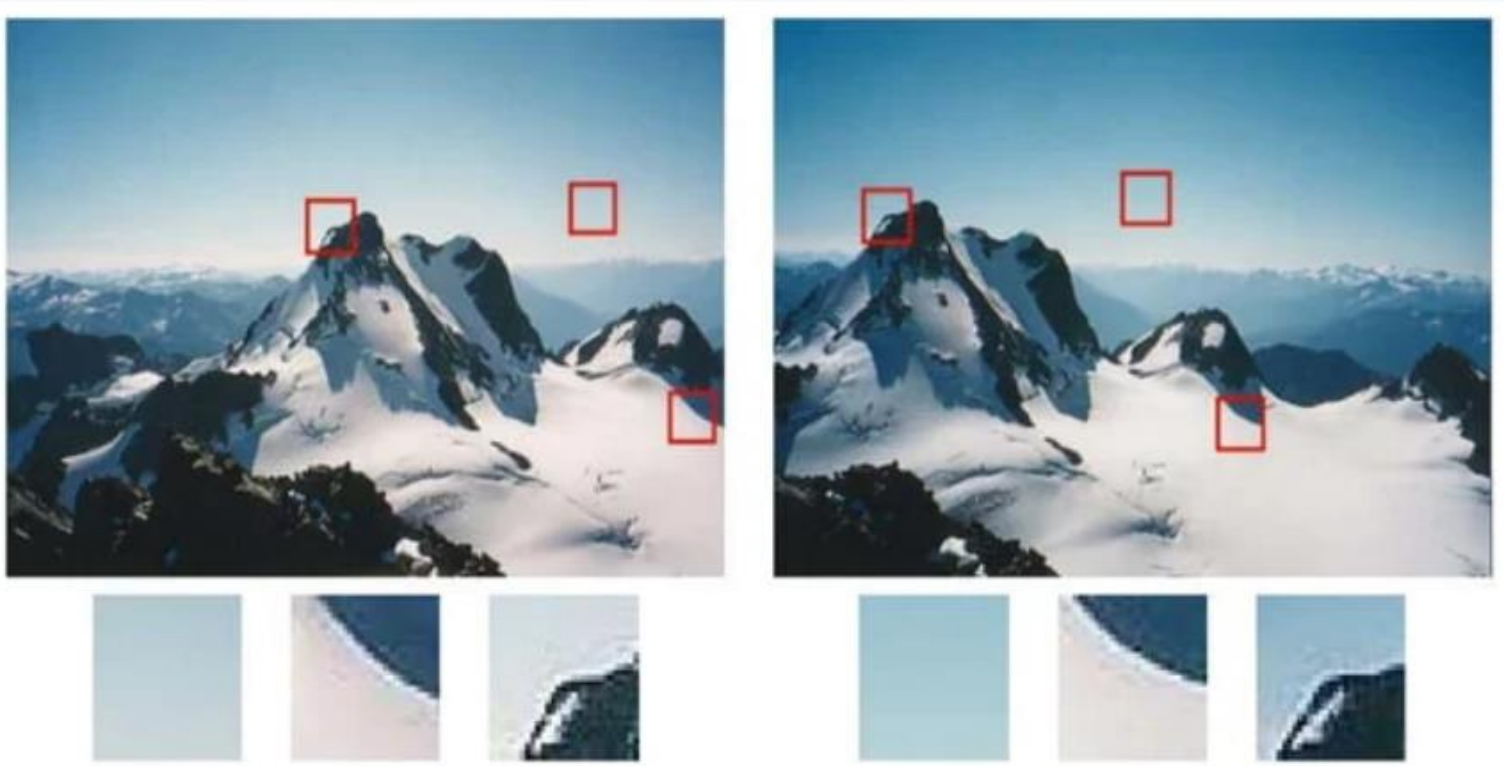
- Finding feature points and their correspondences in 2 approaches:

(1) Find features in one image that is accurately tracked using a local search technique, such as correlation or least squares.
  - More suitable when images are taken from nearby viewpoints or in rapid succession.
  - E.g: video sequences.

(2) Independently detect features in all the images under consideration and then match features based on their local appearance.
  - More suitable when a large amount of motion or appearance change is expected.
  - E.g: stitching together panoramas, establishing correspondences in wide baseline stereo (Estimation of fundamental matrix), or performing object recognition.

Feature Detectors

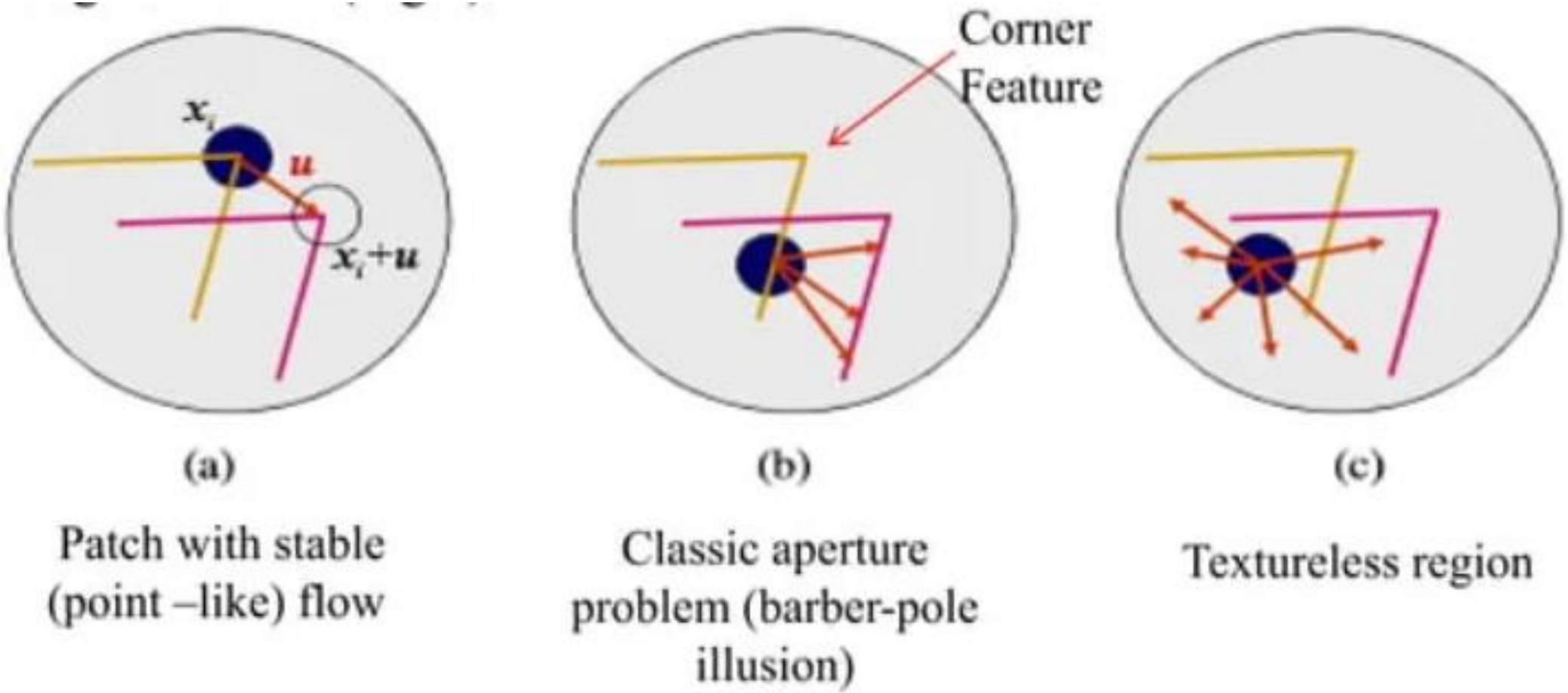
- Find the good features for matching with other images:
  - Figure shows with three sample patches in that how well they might be matched or tracked.
  - From the figure observe, textureless patches are nearly impossible to localize. Patches with large contrast changes (gradients) are easier to localize.



Feature Detectors

- The two images I0 (yellow) and I1 (red) are overlaid.
- The red color vector ‘u’ indicates the displacement between the patch centers.
- W(xi) weighting function (patch window) is shown as a dark circle.
- Patches with gradients in atleast two (significantly) different orientations are the easiest to localize (Fig a).
- Although straight line segments at a single orientation suffer from the aperture problem i.e., it is only possible to align the patches along the direction normal to the edge direction (Fig b).

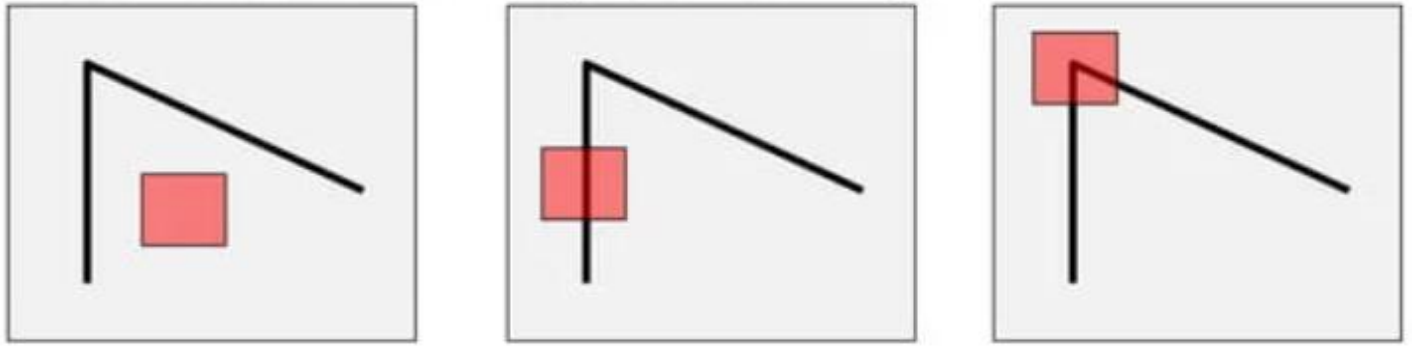
Feature Detectors



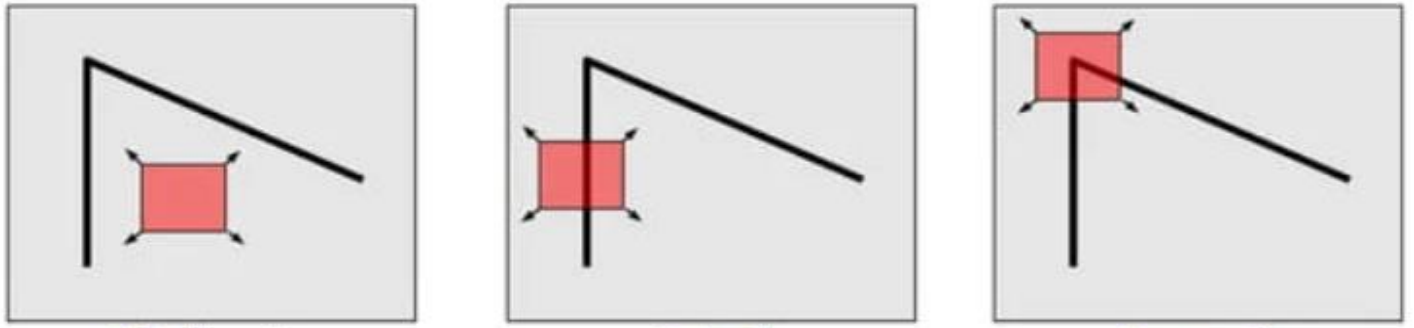
Feature Detectors

- Find unique features i.e., look for unusual regions in images.

(a) Consider the small window of pixels in the image:



(b) 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

Feature Detection

- Edges: Edges are points where there is a boundary (or an edge) between two image regions. In general, an edge can be of almost arbitrary shape and may include junctions.

- Corners: Corners are another unique part of an image, As compared to edges, corners are easier to detect since maximum change in the intensity pixels occur at the corners because it includes atleast two edges and hence the change in intensity of the pixels is higher in corner as compared to edges.

Feature Detection

- Types of Edges:

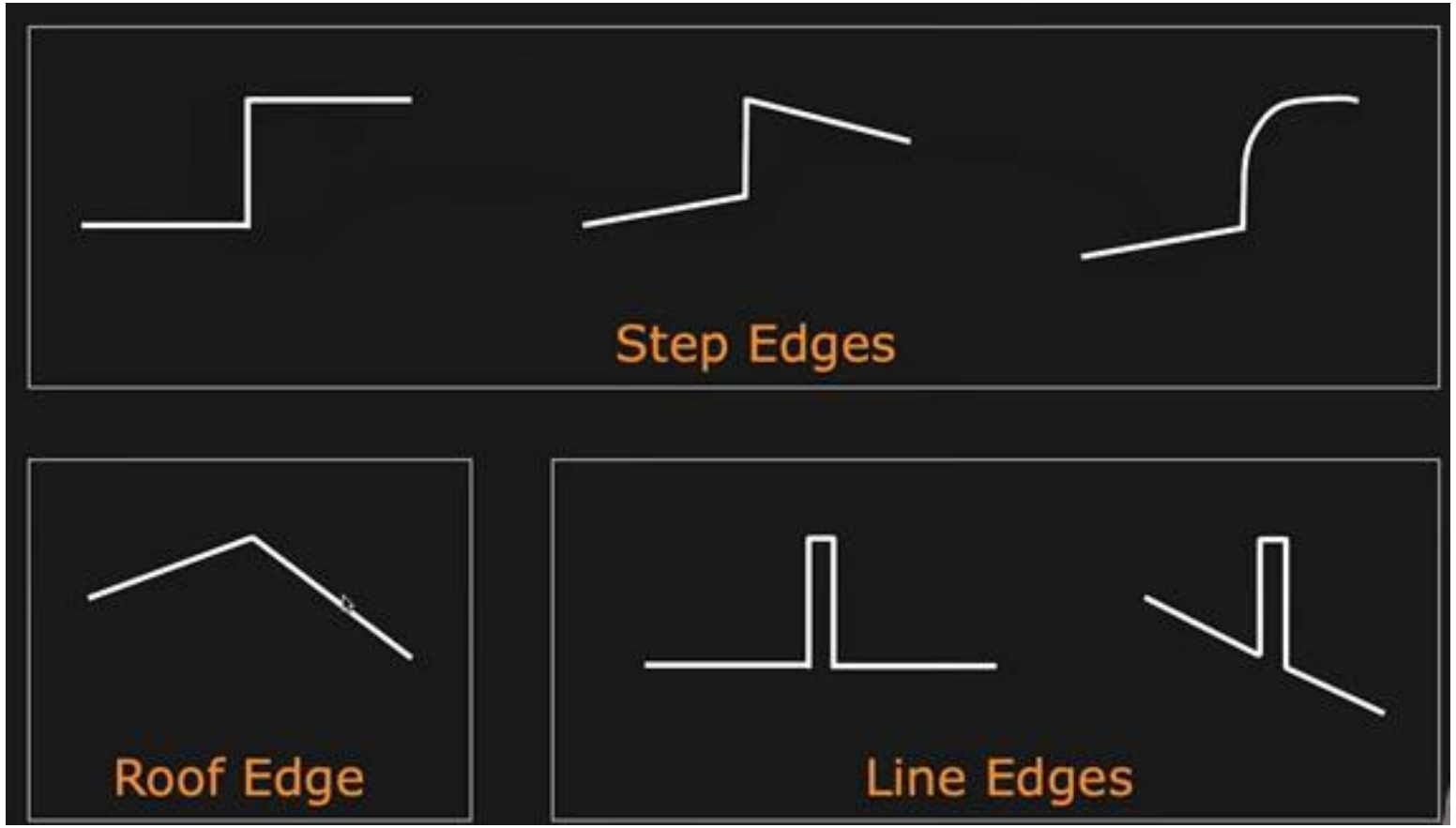


Image Features

- For example: if we give you a feature like a wheel and ask you to guess whether the object is a motorcycle or a dog. What would your guess be? Answer: A motorcycle is correct.!!!

  - In this case, the wheel is a strong feature that clearly distinguishes between motorcycles and dogs.

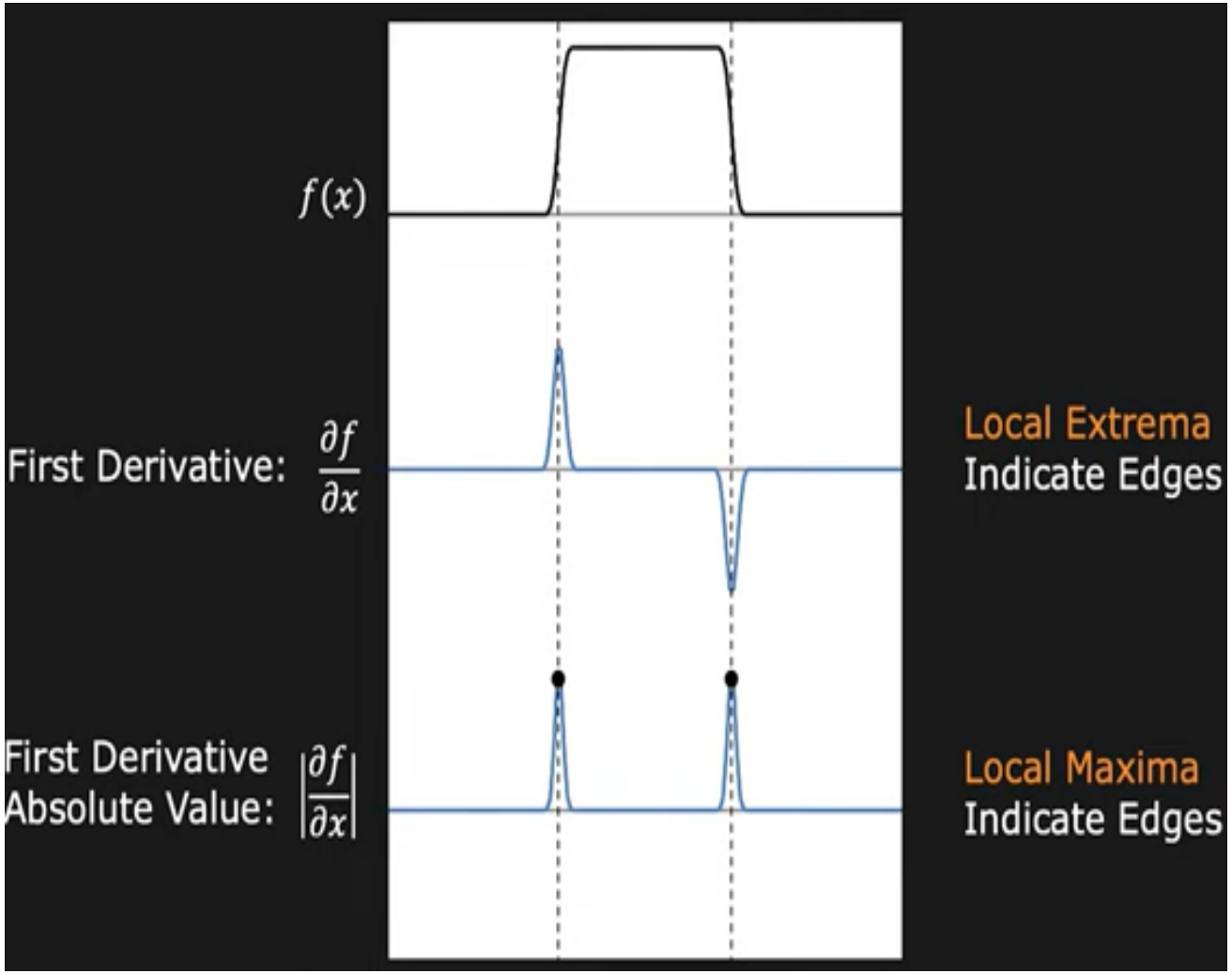
- If we give you the same feature (a wheel) and ask you to guess whether the object is a bicycle or a motorcycle. What would your guess be?

  - In this case, this feature isn't strong enough to distinguish between both objects. Then we need to look for more features like a mirror, license plate, may be a pedal that collectively describes an object etc.

Feature Detection Techniques

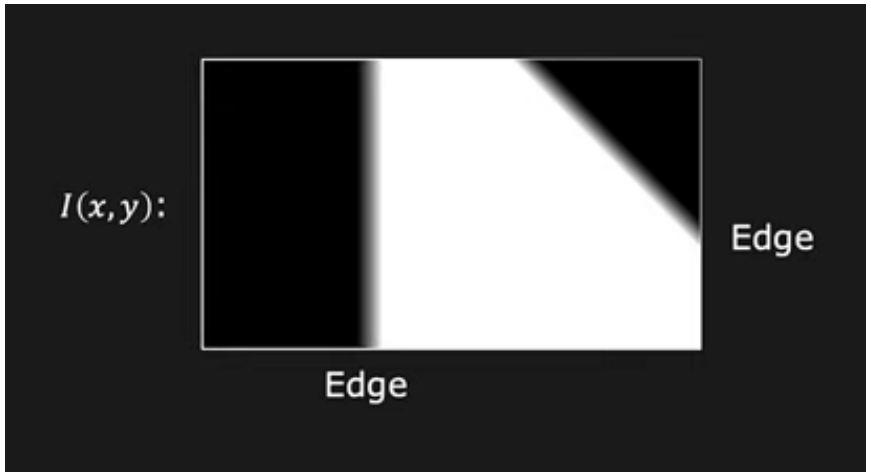
- We can convert a 2D image into a set of points where image intensity changes rapidly.
- Techniques used:
  - Edge detection using gradients
  - Edge detection using Laplacian and Gaussian Detector
  - Harris Corner Detection
  - SIFT
  - RANSAC
  - Least Squares
  - Color and Texture and feature based alignment

1D Edge Detection



- Basic Calculus: Derivative of a continuous function represents the amount of change in the function.

2D Edge Detection



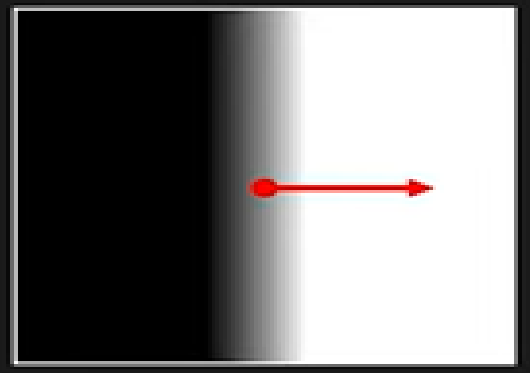
- Basic Calculus: Partial derivatives of a 2D function represents the amount of change along each dimensions.

Gradient (∇)

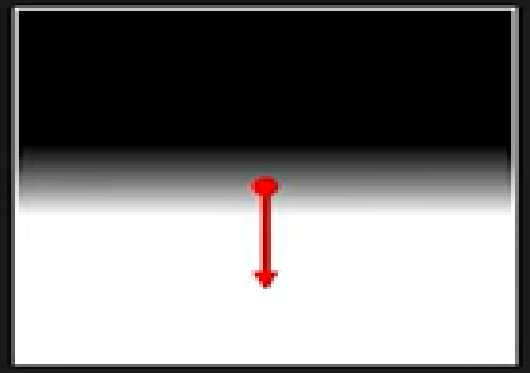
Gradient (Partial Derivatives) represents the direction of most rapid change in intensity

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

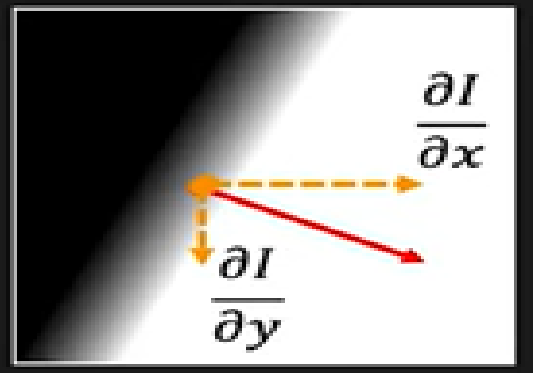
Pronounced as "Del I"



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



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

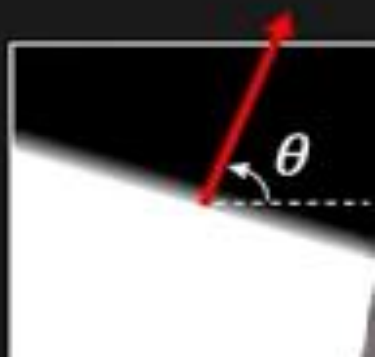


$$\nabla I = \left[\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right]$$

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

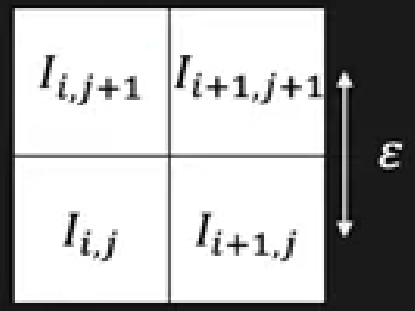


Discrete Gradient (∇) Operator

Finite difference approximations:

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

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



Can be implemented as Convolution!

$$\frac{\partial}{\partial x} \approx \frac{1}{2\varepsilon}\begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \qquad \frac{\partial}{\partial y} \approx \frac{1}{2\varepsilon}\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$$

Note: Convolution flips have been applied

Comparing Gradient (∇) Operators

Gradient	Roberts	Prewitt	Sobel (3x3)	Sobel (5x5)
$\frac{\partial I}{\partial x}$	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -2 & 0 & 2 & 1 \\ -2 & -3 & 0 & 3 & 2 \\ -3 & -5 & 0 & 5 & 3 \\ -2 & -3 & 0 & 3 & 2 \\ -1 & -2 & 0 & 2 & 1 \end{bmatrix}$
$\frac{\partial I}{\partial y}$	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 3 & 5 & 3 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ -2 & -3 & -5 & -3 & -2 \\ -1 & -2 & -3 & -2 & -1 \end{bmatrix}$

Good Localization → Poor Localizatio
Noise Sensitive → Less Noise Sensitiv
Poor Detection → Good Detectio

Edge Thresholding

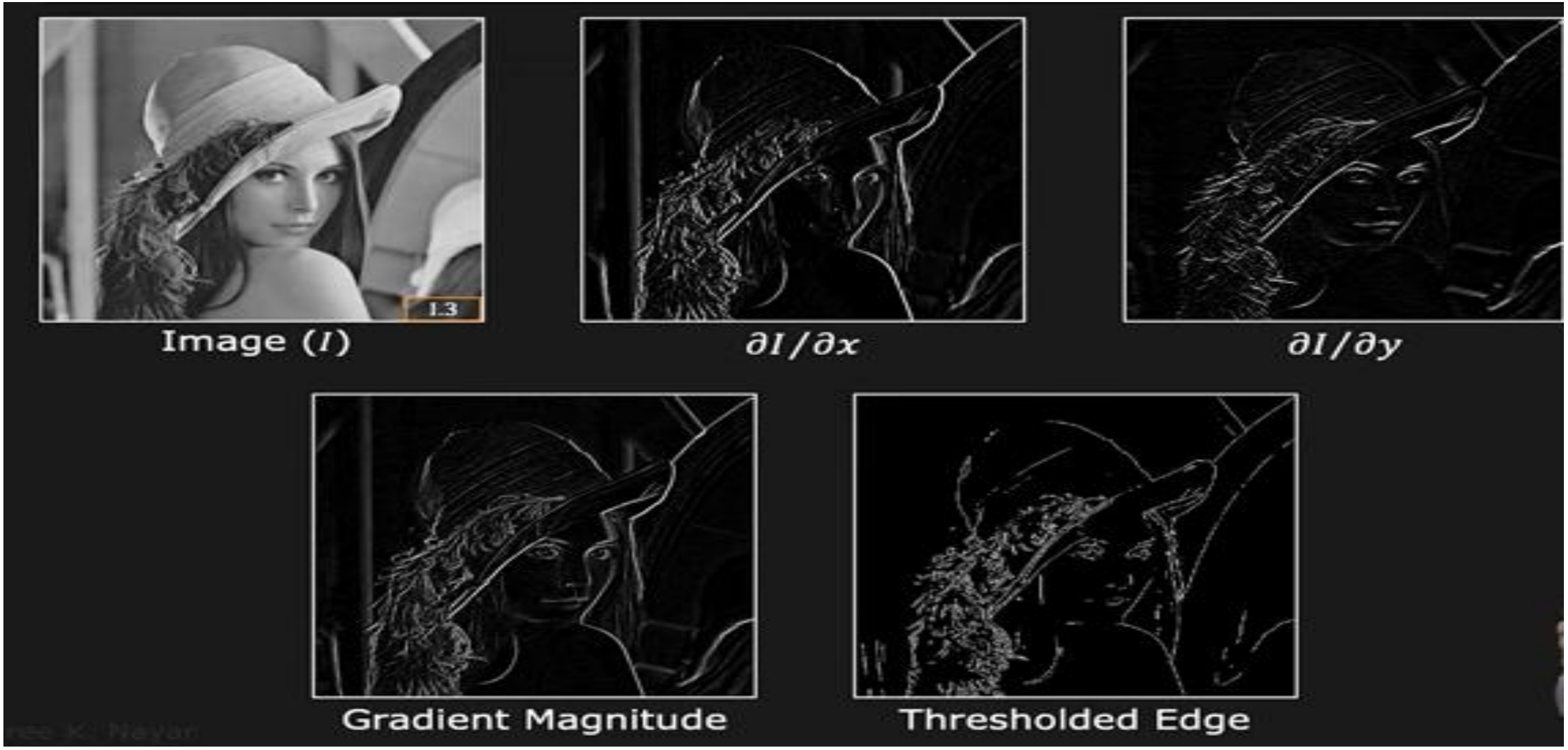
Standard: (Single Threshold T)

$\|\nabla I(x,y)\| < T$	Definitely Not an Edge
$\|\nabla I(x,y)\| \geq T$	Definitely an Edge

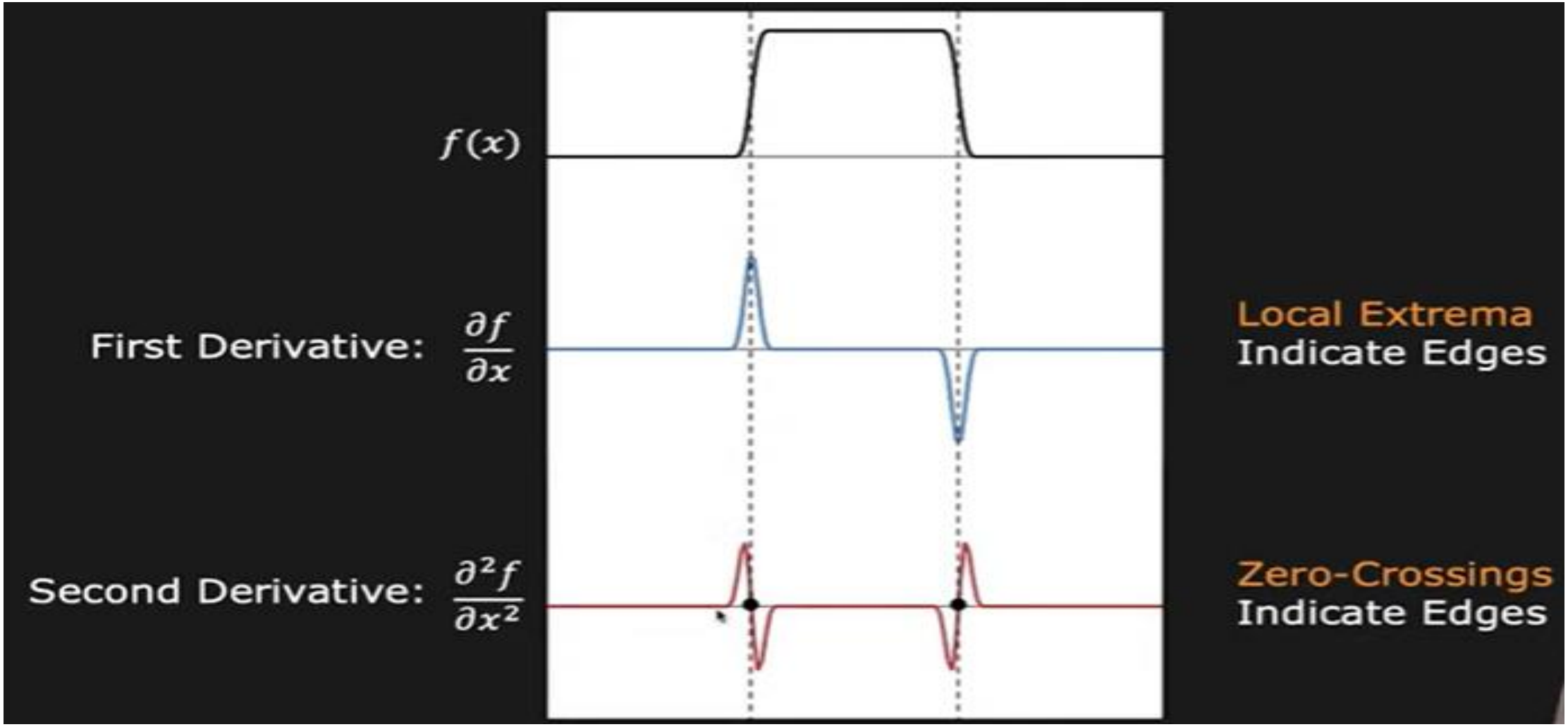
Hysteresis Based: (Two Thresholds $T_0 < T_1$)

$\|\nabla I(x,y)\| < T_0$	Definitely Not an Edge
$\|\nabla I(x,y)\| \geq T_1$	Definitely an Edge
$T_0 \leq \|\nabla I(x,y)\| < T_1$	Is an Edge if a Neighboring Pixel is Definitely an Edge

Sobel Edge Detector



Edge detection using Laplacian



Laplacian (∇^2) as Edge Detector

Laplacian: Sum of Pure Second Derivatives

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

Pronounced as "Del Square I"

- Edges are "zero-crossings" in Laplacian of image
- Laplacian does not provide directions of edges

Discrete Laplacian(∇^2) Operator

Finite difference approximations:

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

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

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

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

ε

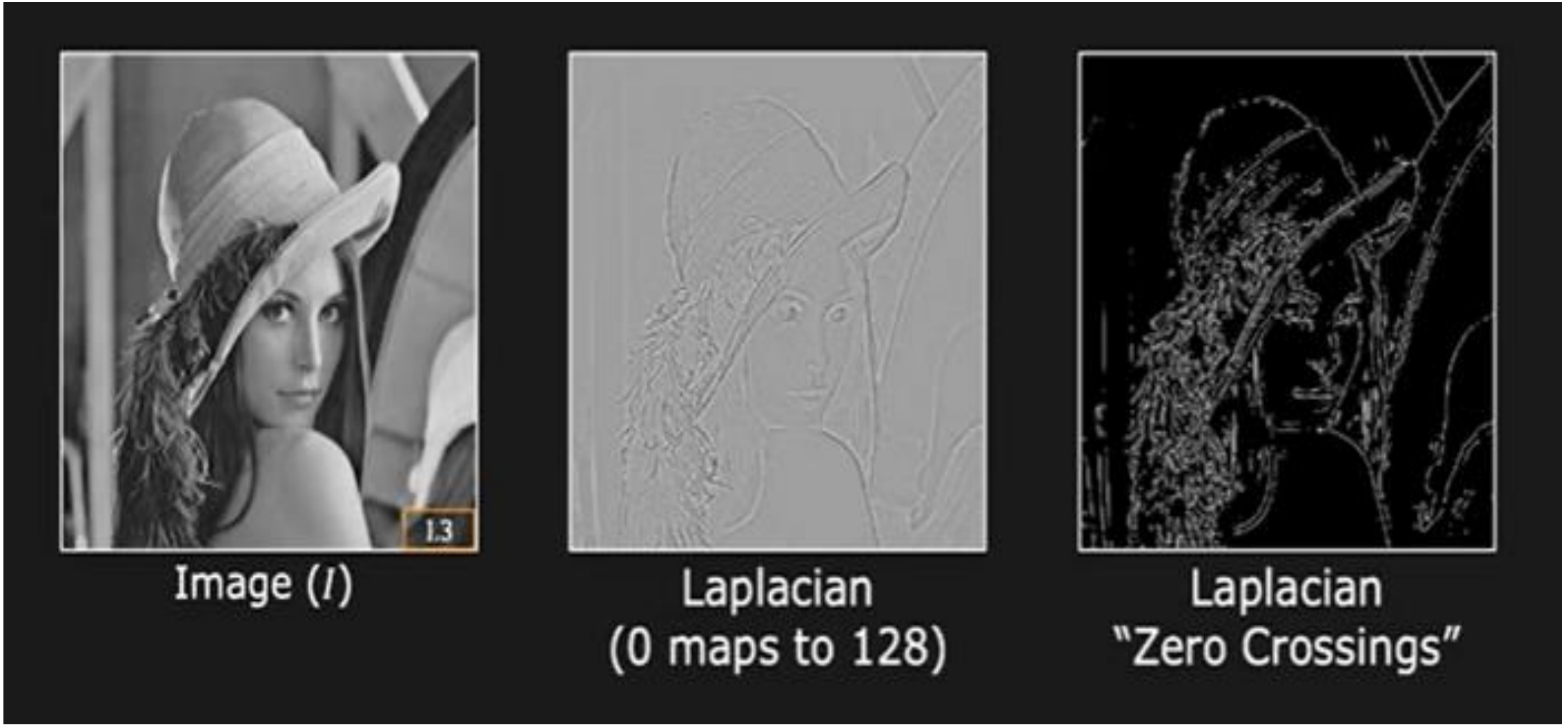
Convolution Mask:

$$\nabla^2 \approx \frac{1}{\varepsilon^2}\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \text{OR} \quad \nabla^2 \approx \frac{1}{6\varepsilon^2}\begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix}$$

(More Accurate

hree K. Nayar

Laplacian Edge Detector



Harries Corner Detector

- This approach gives a mathematical approach to detect a corner in an image.
- We know that the shift in (u,v) is represented by:

$$\sum [I(x+u, y+v) - I(x,y)]^2$$

- When we rewrite the equation on matrix form:

$$\sum \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

How do we use it?

- We now have the equation where W is windowing function (usually 1)

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

- We further calculate the trace and determinant of M to find R.
- K is empirically determined constant -

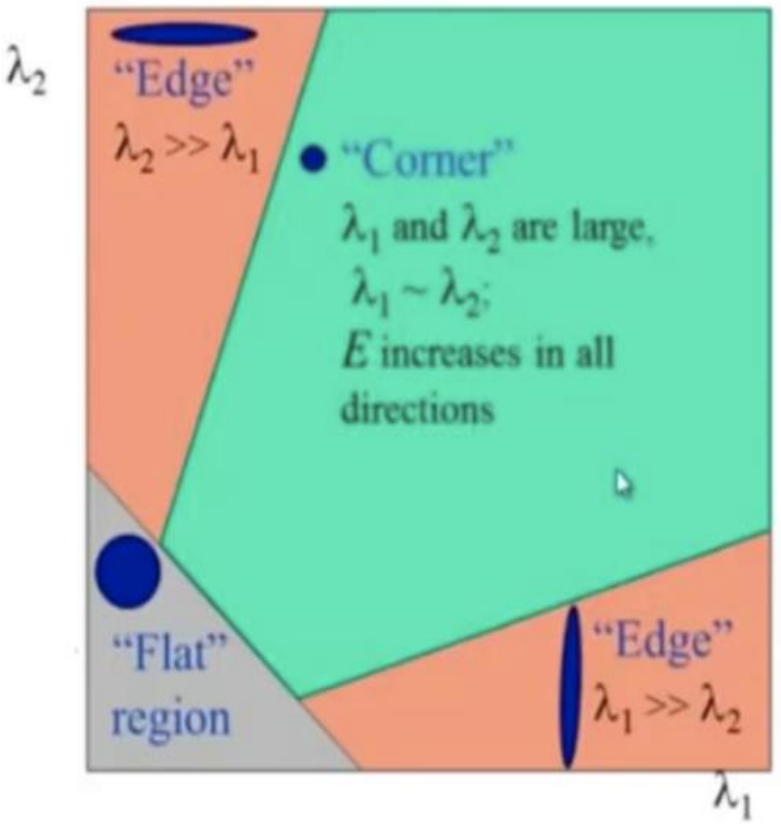
$$R = \det M - k(trace\, M)^2$$

$$\det M = \lambda_1 \lambda_2$$

$$trace\, M = \lambda_1 + \lambda_2$$

How to know if it is a corner?

- The value of R will determine it if is a corner or an edge
- If it is a corner value of R will be positive and large
- If it is an edge value of R will be negative and large in magnitude.
- Magnitude of R will be small for a flat region
- Thus we say that eigenvalues of M determine if it a corner or not, usually when both the eigenvalues are larger it a corner.



Harries Corner Detector

1. |R| is small, which happens when λ1 and λ2 are small, the region is flat.
2. R<0, which happens when λ1>>λ2 or vice versa, the region is edge.
3. R is large, which happens when λ1 and λ2 are large and λ1~λ2, the region is a corner.

Harries Corner Detection – Example Problem

Input Image:

0	0	1	4	9
1	0	5	7	11
1	4	9	12	16
3	8	11	14	16
8	10	15	16	20

differentiation kernels:

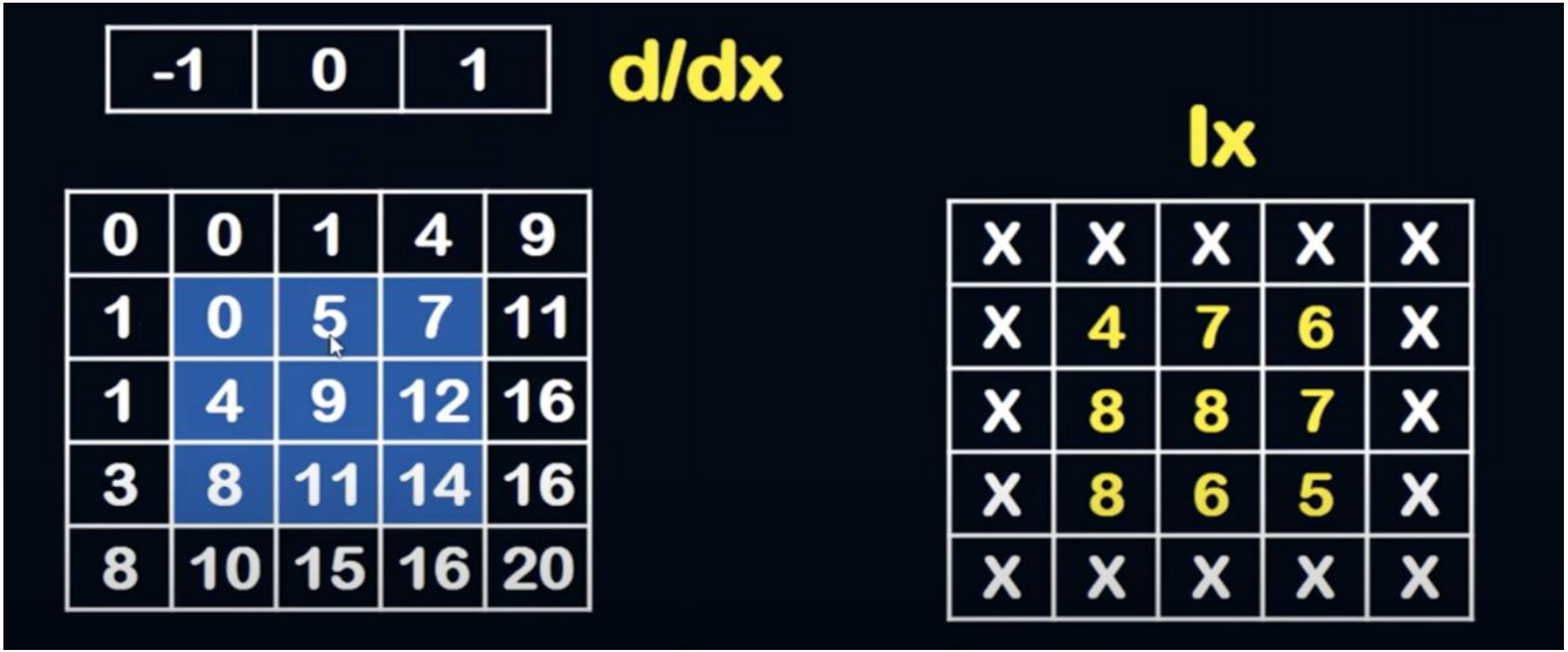
-1	0	1

d/dx

-1
0
1

d/dy

- X-Axis:


-1
0
1
d/dx
0 0 1 4 9
1 0 5 7 11
1 4 9 12 16
3 8 11 14 16
8 10 15 16 20
Ix
X X X X X
X 4 7 6 X
X 8 8 7 X
X 8 6 5 X
X X X X X

- Y-Axis:

-1
0
1

d/dy

0	0	1	4	9
1	0	5	7	11
1	4	9	12	16
3	8	11	14	16
8	10	15	16	20

Iy

X	X	X	X	X
X	4	8	8	X
X	8	6	7	X
X	6	6	4	X
X	X	X	X	X

Ix

X	X	X	X	X
X	4	7	6	X
X	8	8	7	X
X	8	6	5	X
X	X	X	X	X

Iy

X	X	X	X	X
X	4	8	8	X
X	8	6	7	X
X	6	6	4	X
X	X	X	X	X

$$4^2 + 7^2 + 6^2 + 8^2 + 8^2 + 7^2 + 8^2 + 6^2 + 5^2 = 403$$

$$4^2 + 8^2 + 8^2 + 8^2 + 6^2 + 7^2 + 6^2 + 6^2 + 4^2 = 381$$

$$4 * 4 + 7 * 8 + 6 * 8 + 8 * 8 + 8 * 6 + 7 * 7 + 8 * 6 + 6 * 6 + 5 * 4 = 385$$

$$H = \begin{bmatrix} 403 & 385 \\ 385 & 381 \end{bmatrix}$$

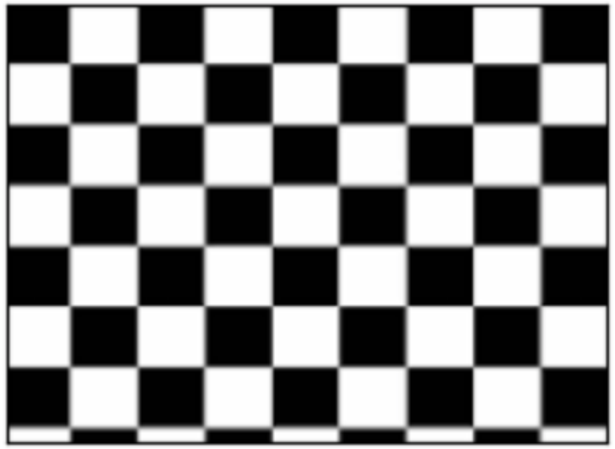
$$H = \begin{bmatrix} 403 & 385 \\ 385 & 381 \end{bmatrix}$$

$$C = \det(H) - k\,\mathrm{trace}(H)^2$$

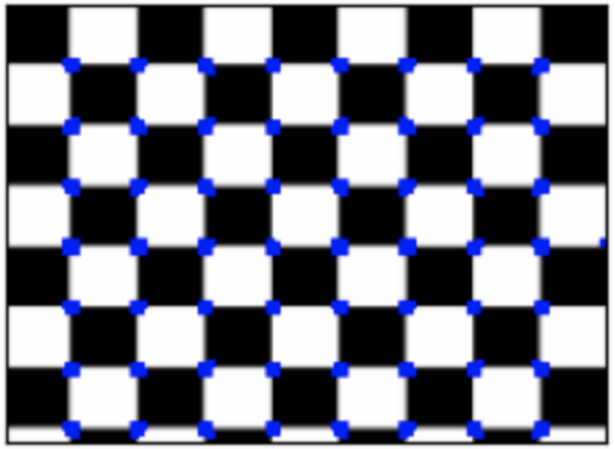
$$C = 5318 - 0.04 * (784)^2 = -19268.24$$

If C is large: Corner
If C is negative: Edge
If |C| is small: flat

Original Image



Harries Corner Detector



Original Image

Harries Corner Detector

SIFT – Scale Invariant Feature Transform

SIFT algorithm helps locate the local features in an image, commonly known as the '*keypoints*' of the image. These keypoints are scale & rotation invariants that can be used for various computer vision applications, like image matching, object detection, scene detection, etc.

Goal

- Extracting distinctive invariant features
  - Correctly matched against a large database features from many images
- Invariance to image scale and rotation
- Robustness to
  - Affine distortion,
  - Change in 3D viewpoint,
  - Addition of noise,
  - Change in illumination.

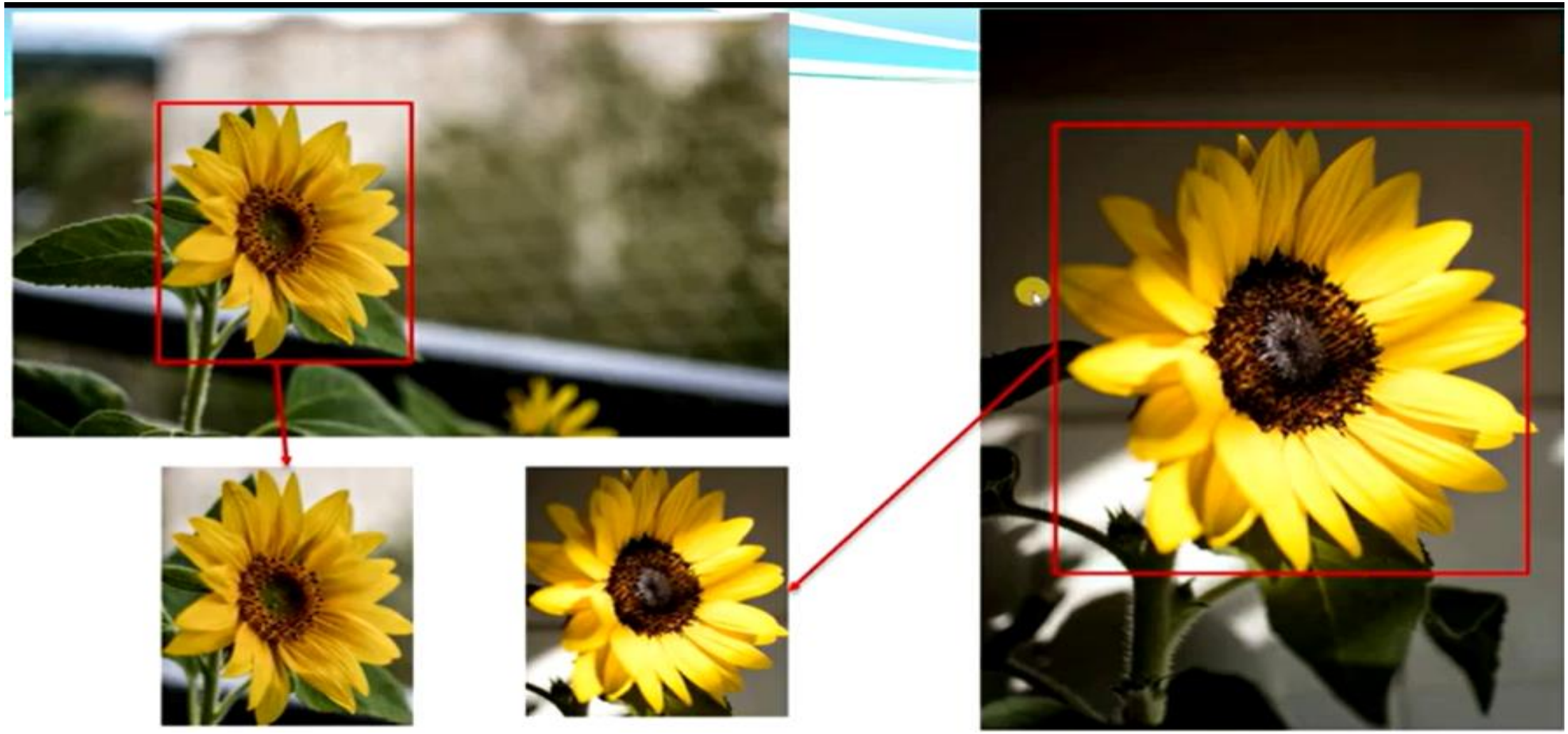
Advantages

- **Locality:** features are local, so robust to occlusion and clutter
- **Distinctiveness:** individual features can be matched to a large database of objects
- **Quantity:** many features can be generated for even small objects
- **Efficiency:** close to real-time performance

Steps for Extracting Key Points

- Scale space peak selection
  - Potential locations for finding features
- Key point localization
  - Accurately locating the feature key points
- Orientation Assignment
  - Assigning orientation to the key points
- Key point descriptor
  - Describing the key point as a high dimensional vector

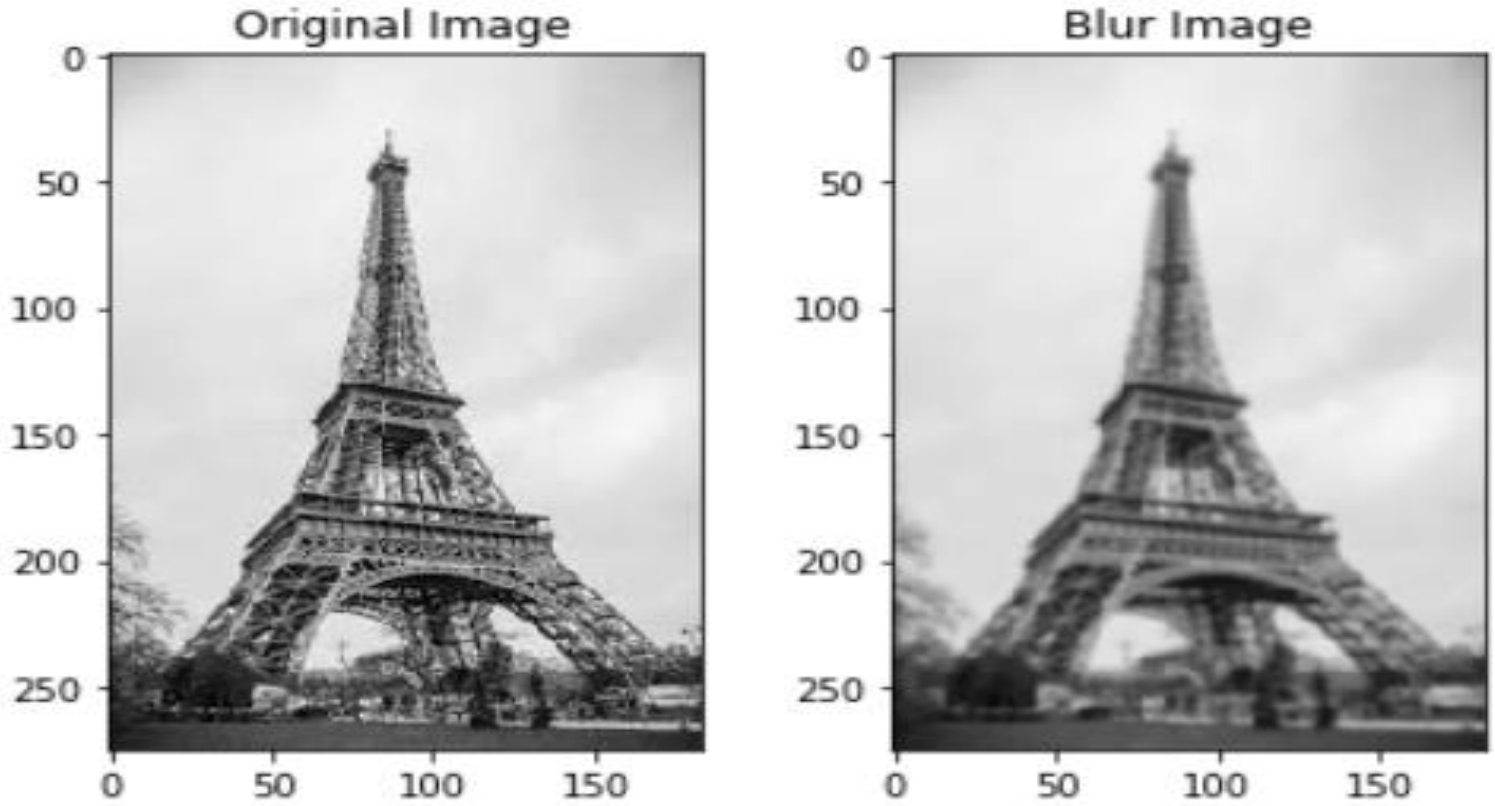
SIFT – Scale Invariant Feature Transform



SIFT – Scale Invariant Feature Transform

SIFT – Scale Invariant Feature Transform

- For every pixel in an image, the Gaussian Blur calculates a value based on its neighboring pixels with a certain sigma value. Below is an example of an image before and after applying the Gaussian Blur.



SIFT – Scale Invariant Feature Transform

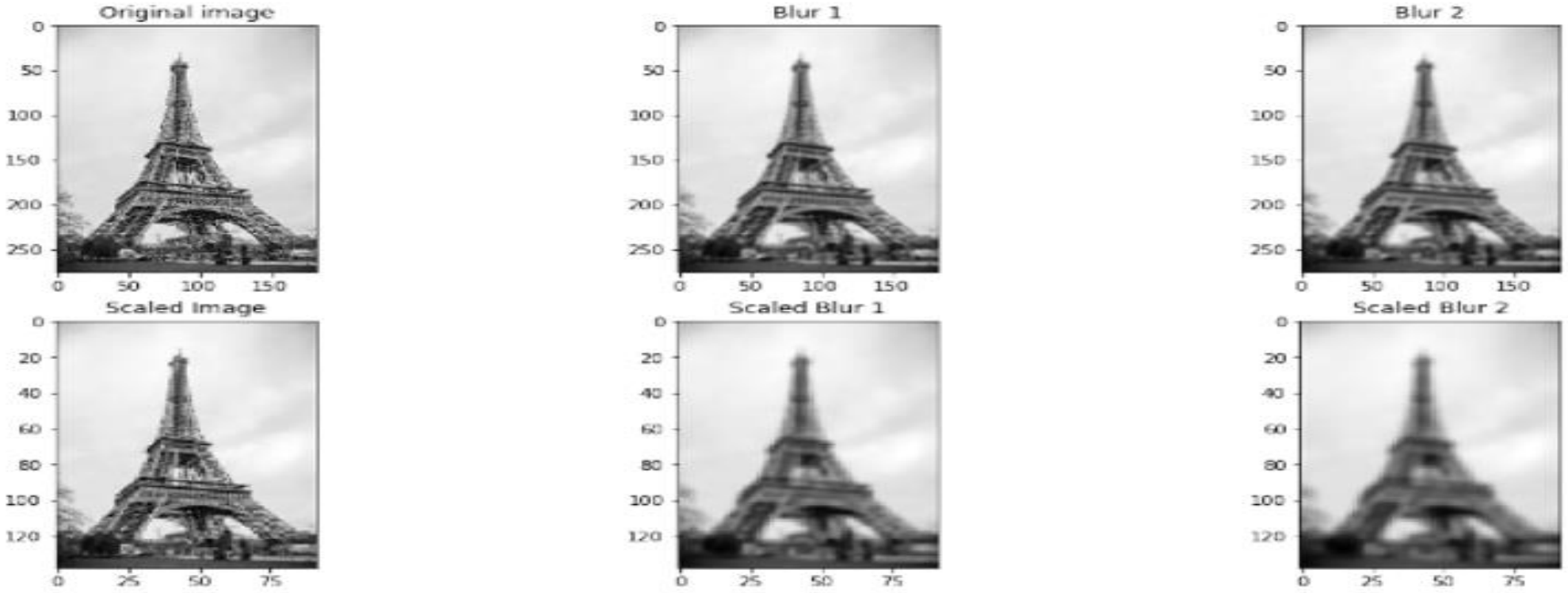
- Gaussian Blur helped in image processing and successfully removed the noise from the images, and we have highlighted the important features of the image. Now, *we need to ensure that these features are scale-dependent.* This means we will be searching for these features on multiple scales by creating a 'scale space'.

- Scale space is a collection of images having different scales, generated from a single image.

- **The major advantage of SIFT features, over-edge features, or hog features is that they are not affected by the size or orientation of the image.**

SIFT – Scale Invariant Feature Transform

Here is an example to understand it in a better manner. We have the original image of size (275, 183) and a scaled image of dimension (138, 92). For both images, two blur images are created:

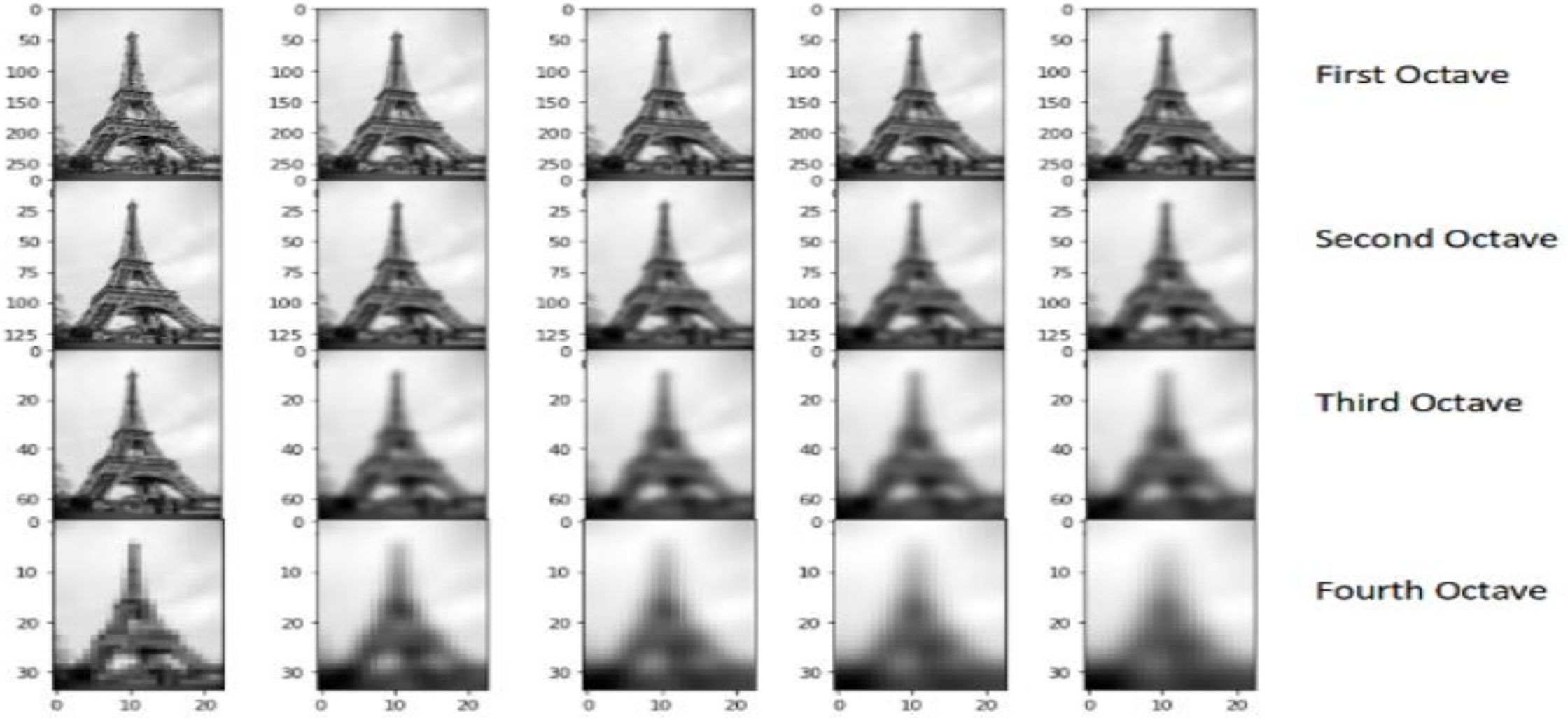


SIFT – Scale Invariant Feature Transform

- how many times do we need to scale the image, and how many subsequent blur images need to be created for each scaled image?

- **The ideal number of octaves should be four**, and for each octave, the number of blur images should be five.

SIFT – Scale Invariant Feature Transform



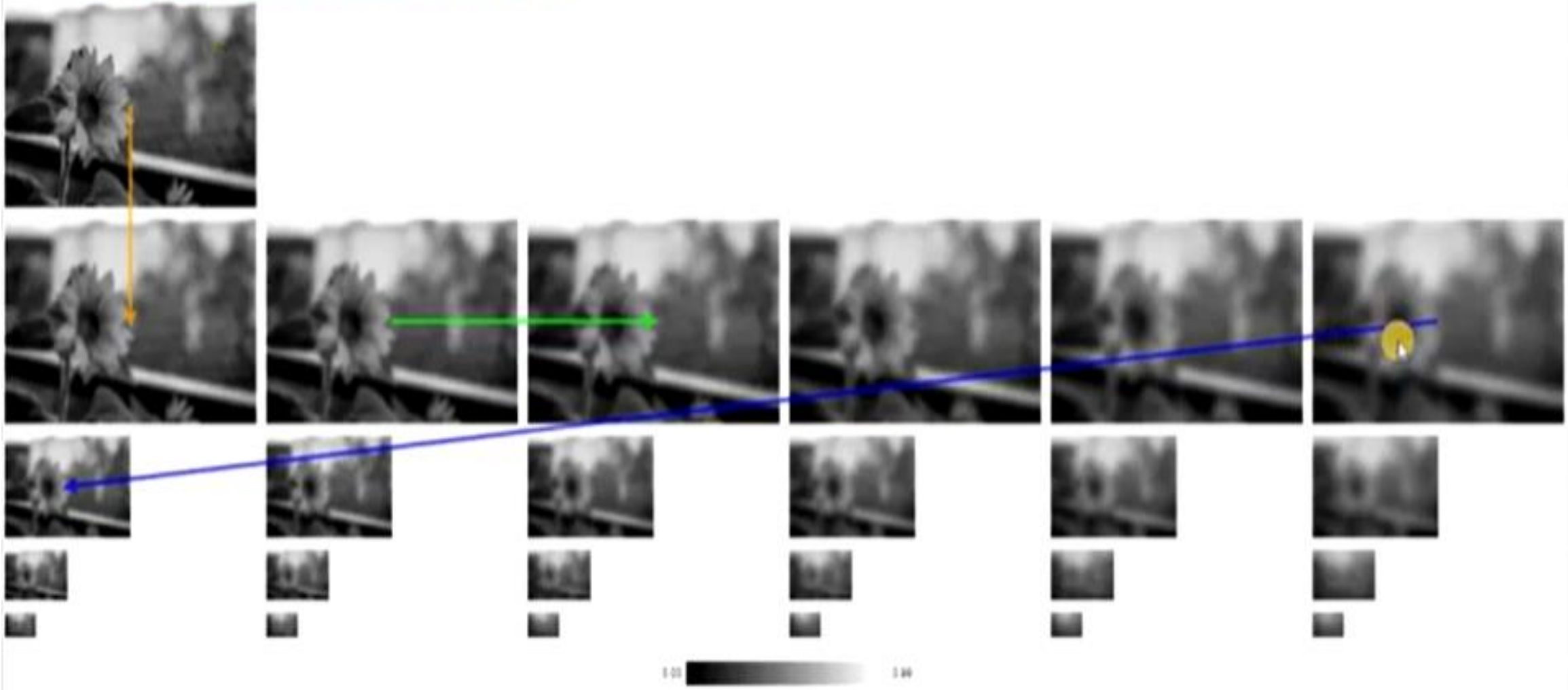
Difference of Gaussian

- we have created images of multiple scales (often represented by σ) and used Gaussian blur for each of them to reduce the noise in the image. Next, we will try to enhance the features using a technique called the Difference of Gaussians or DoG.

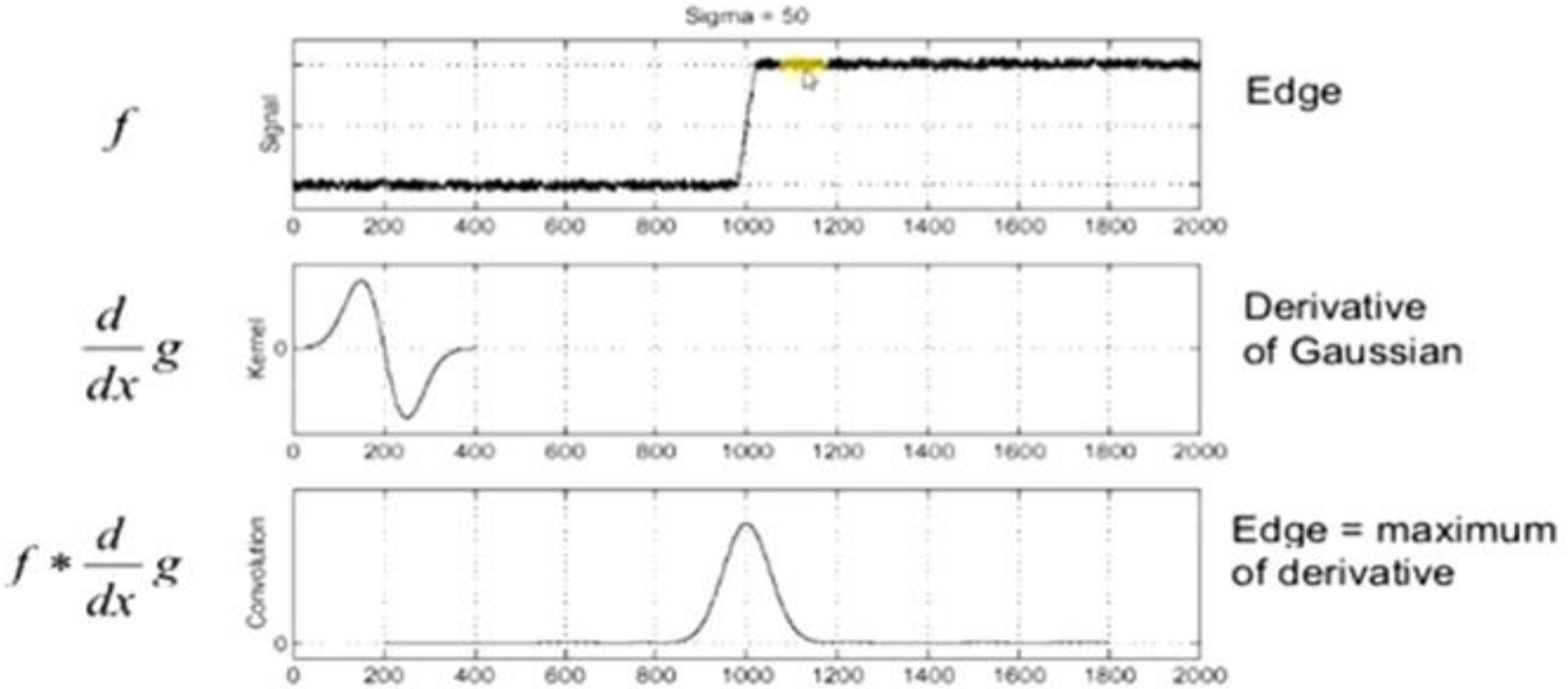
- Difference of Gaussian is a feature enhancement algorithm that involves the subtraction of one blurred version of an original image from another, less blurred version of the original.

- DoG creates another set of images, for each octave, by subtracting every image from the previous image in the same scale.

Blob Detection

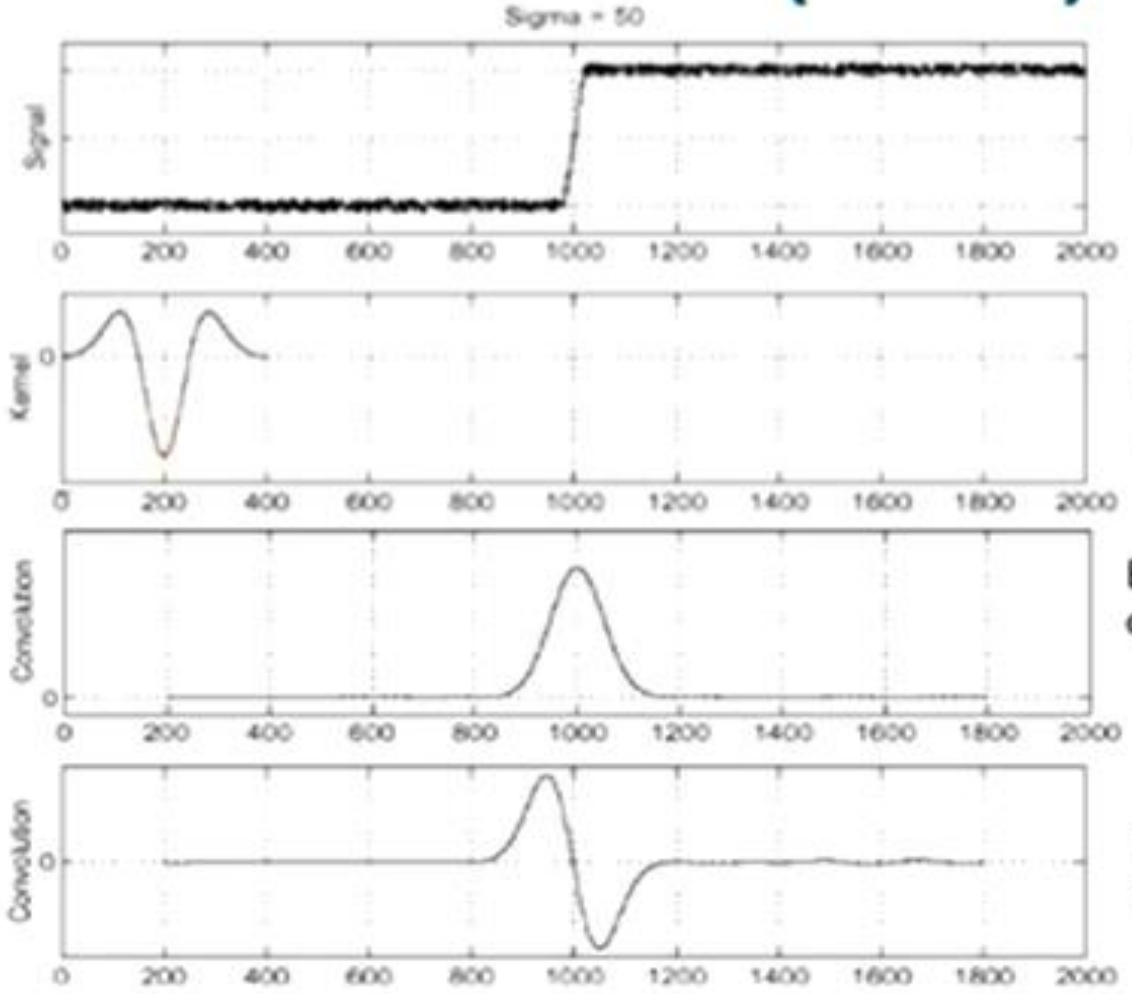


Edge Detection



Edge Detection Take 2 (LoG)

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

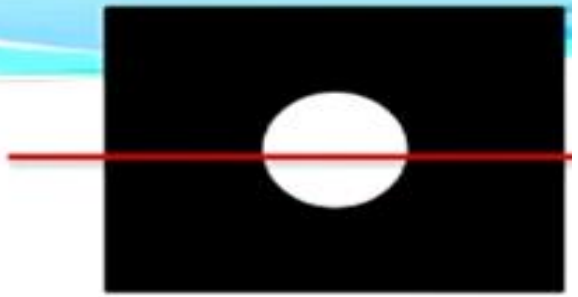


$$f * \frac{d}{dx} g$$

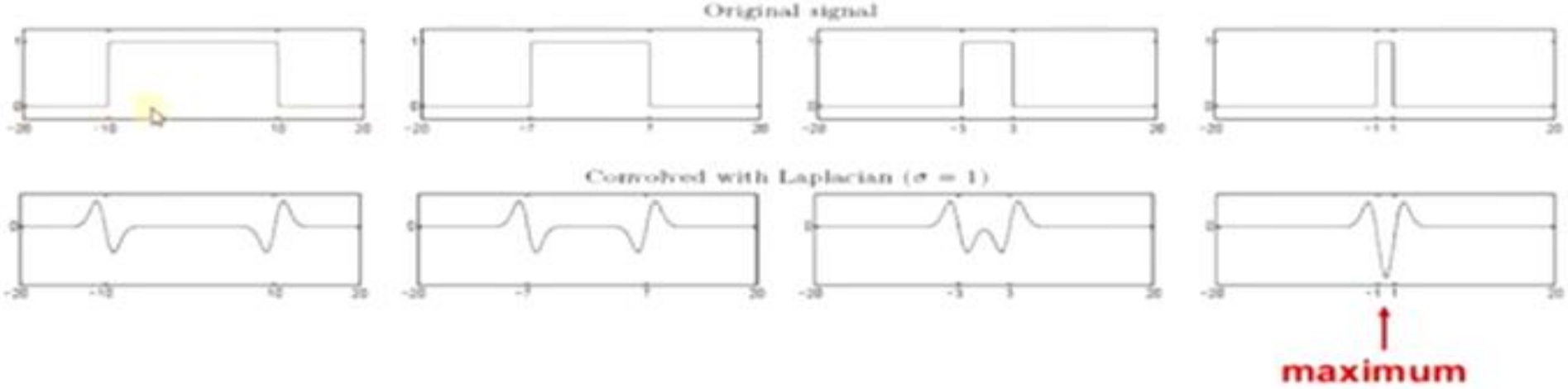
$$f * \frac{d^2}{dx^2} g$$

Image taken

Coming to the point



- Edge = ripple
- Blob = superposition of two ripples

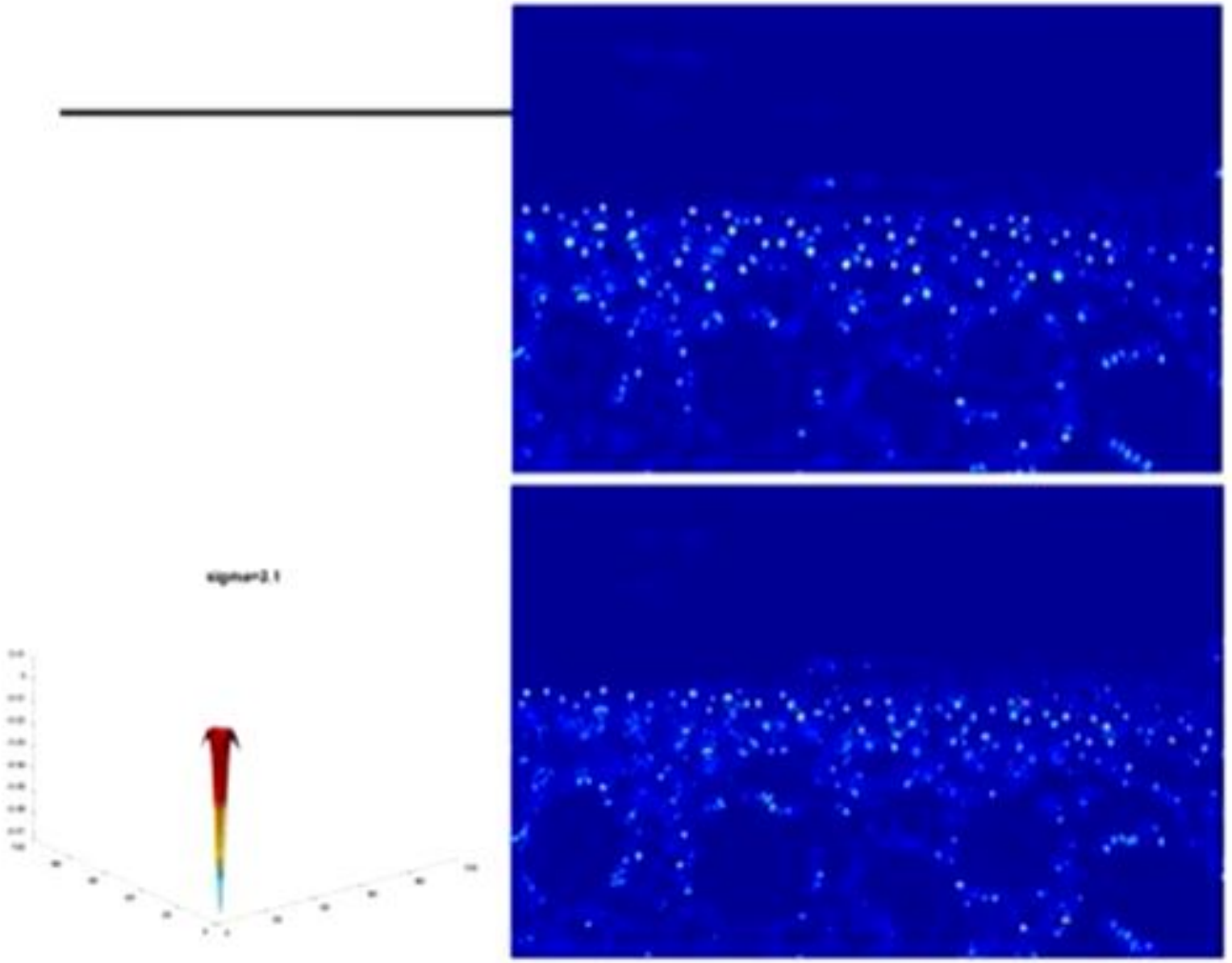


Spatial selection: the magnitude of the Laplacian response will achieve a maximum at the center of the blob, provided the scale of the Laplacian is "matched" to the scale of the blob

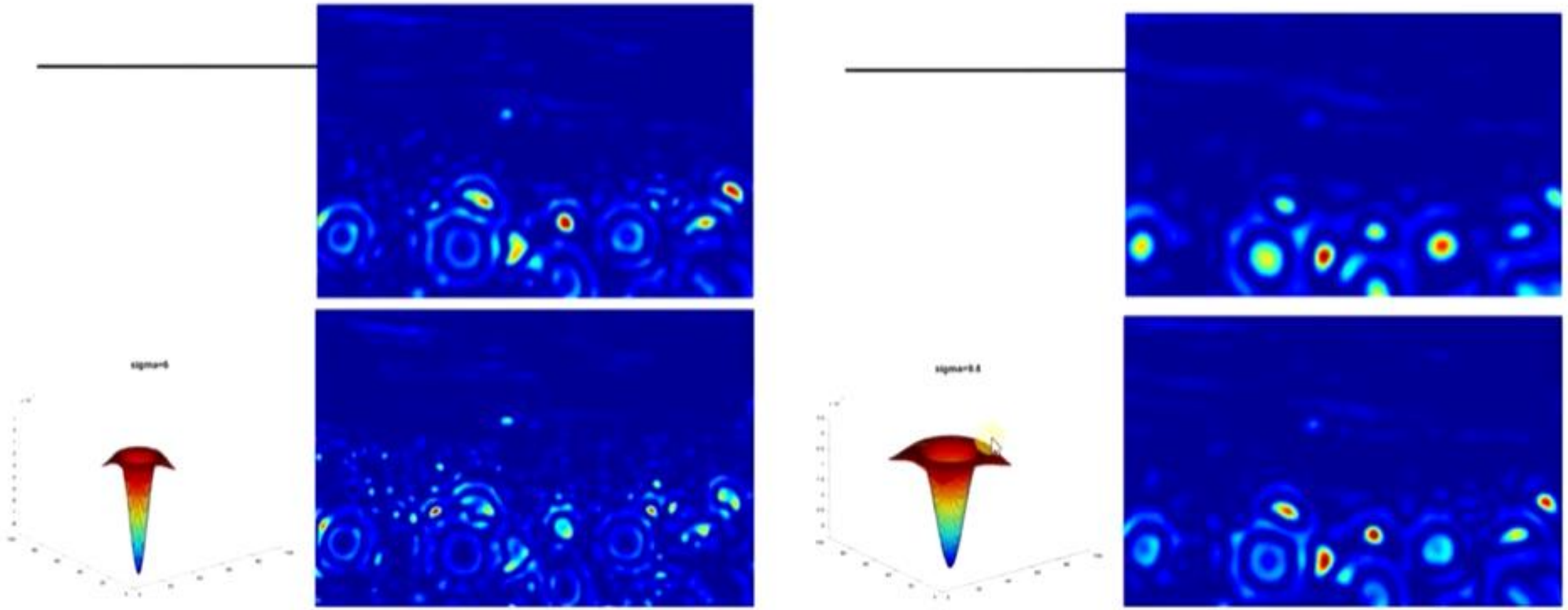
Image taken from here

Blob Detection

Images from Kristen Grauman's slides



Blob Detection



From LoG to DoG

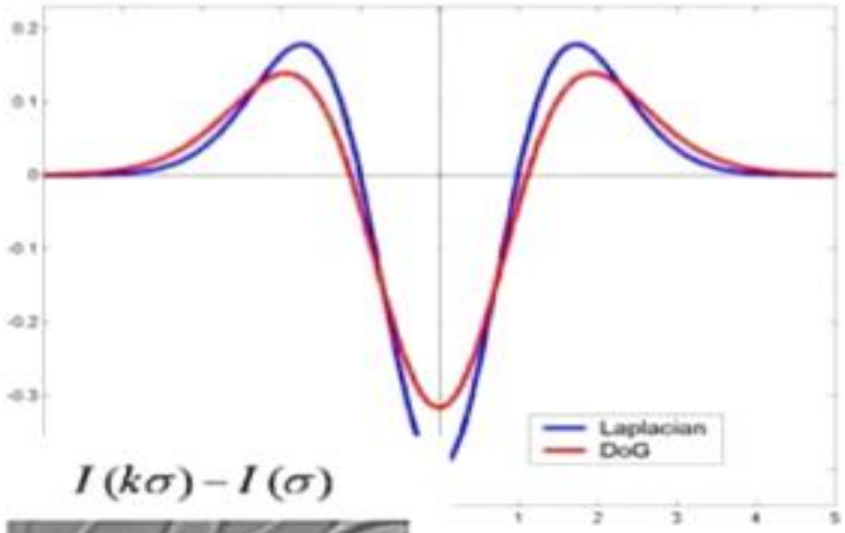
We can approximate the Laplacian with a difference of Gaussians; more efficient to implement.

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

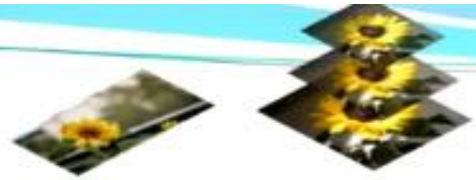
(Laplacian)

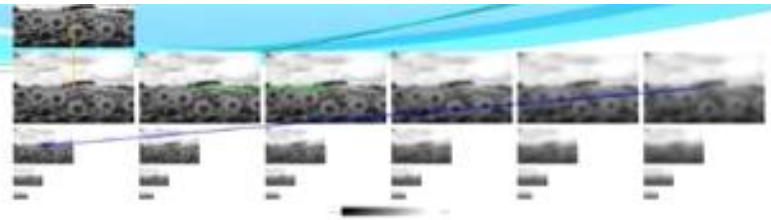
$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

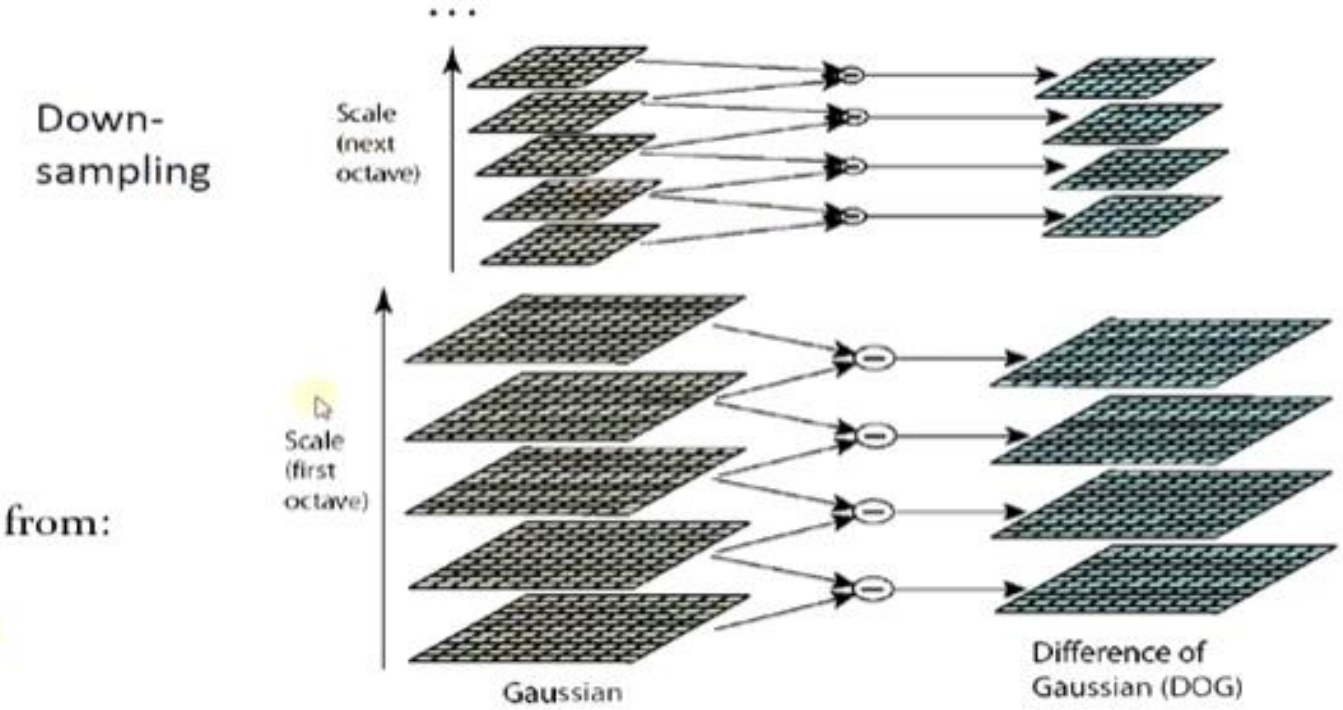
(Difference of Gaussians)



The Scale Space







$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$

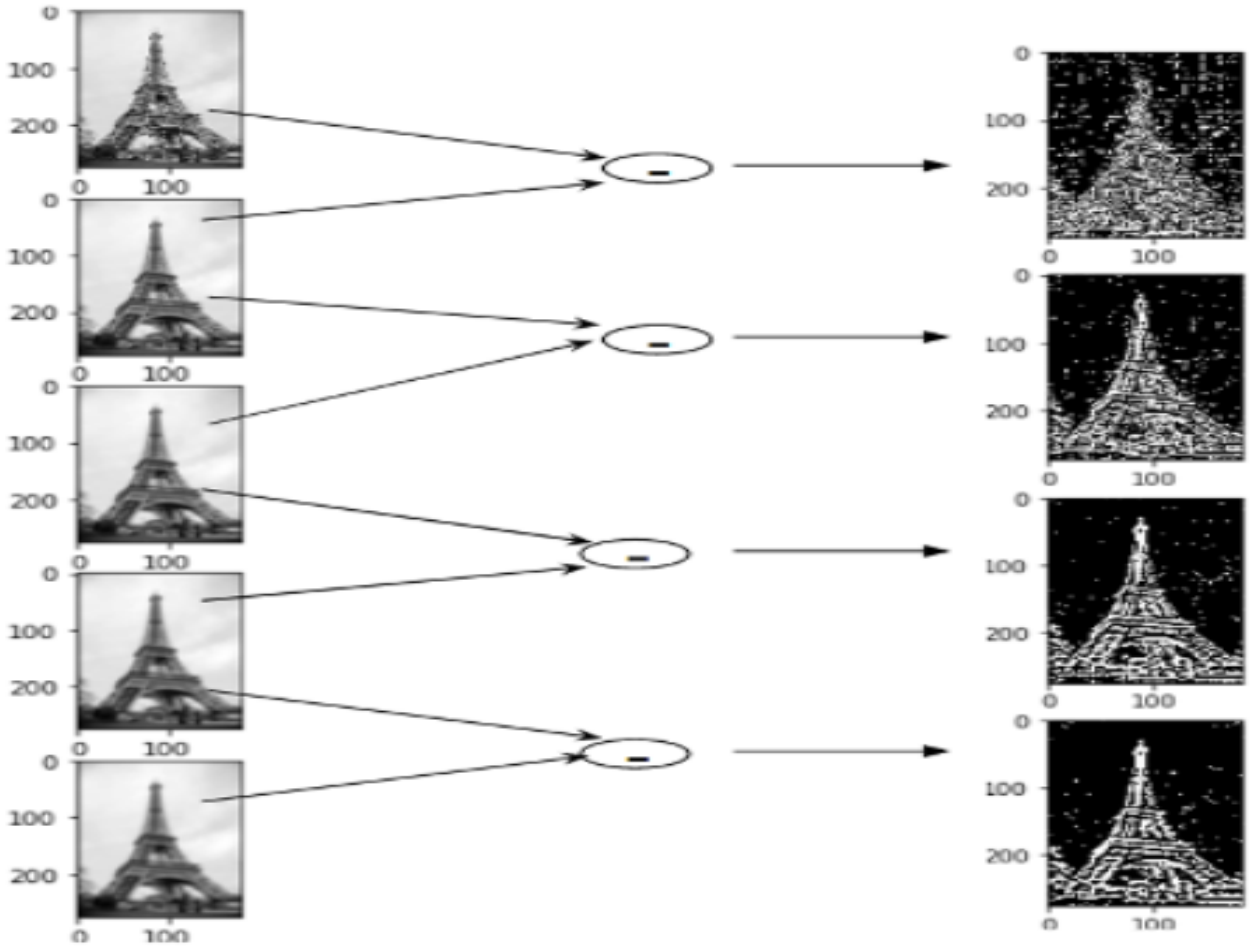
$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

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

ıge taken from:
763
Rajwade

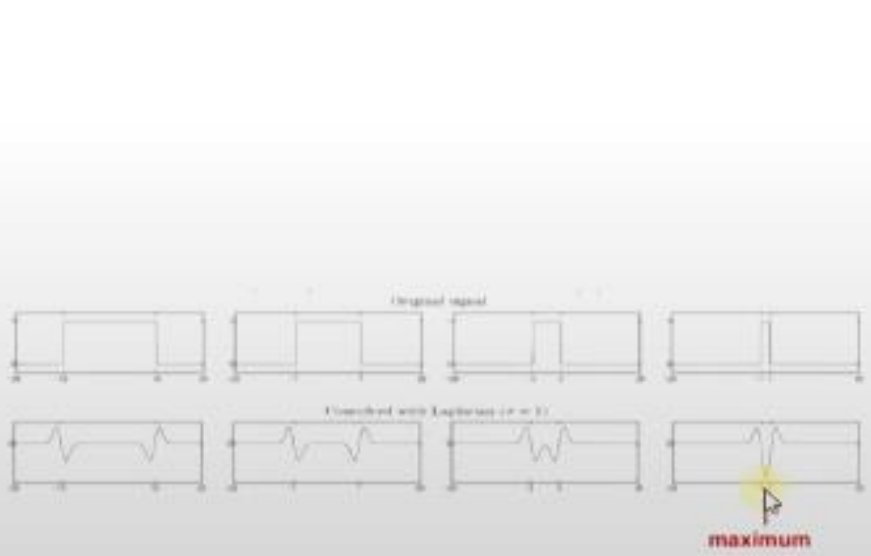
Octave = doubling of σ_0. Within an octave, the adjacent scales differ by a constant factor k. If an octave contains $s+1$ images, then $k = 2^{(1/s)}$. The first image has scale σ_0, the second image has scale $k\sigma_0$, the third image has scale $k^2\sigma_0$, and the last image has scale $k^s\sigma_0$. Such a sequence of images convolved with Gaussians of increasing σ constitute a so-called **scale space**.

On the right, we have four images generated by subtracting the consecutive Gaussians. The results are jaw-dropping!



Scale Space Extrema detection

- Extract local extrema (i.e., minima or maxima) in DoG pyramid.
  - -Compare each point to its 8 neighbors at the same level, 9 neighbors in the level above, and 9 neighbors in the level below (i.e. 26 total).



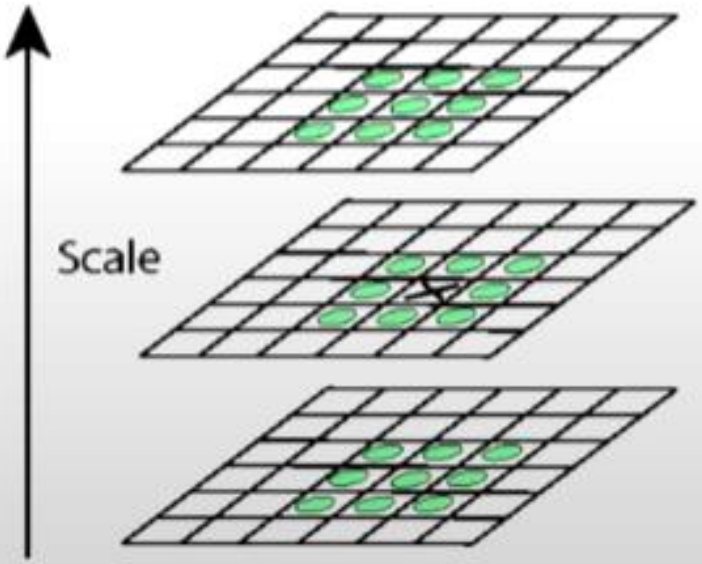


Image taken from:
CS 763
Ajit Rajwade

Initial Points Detection

Feature Point Localization

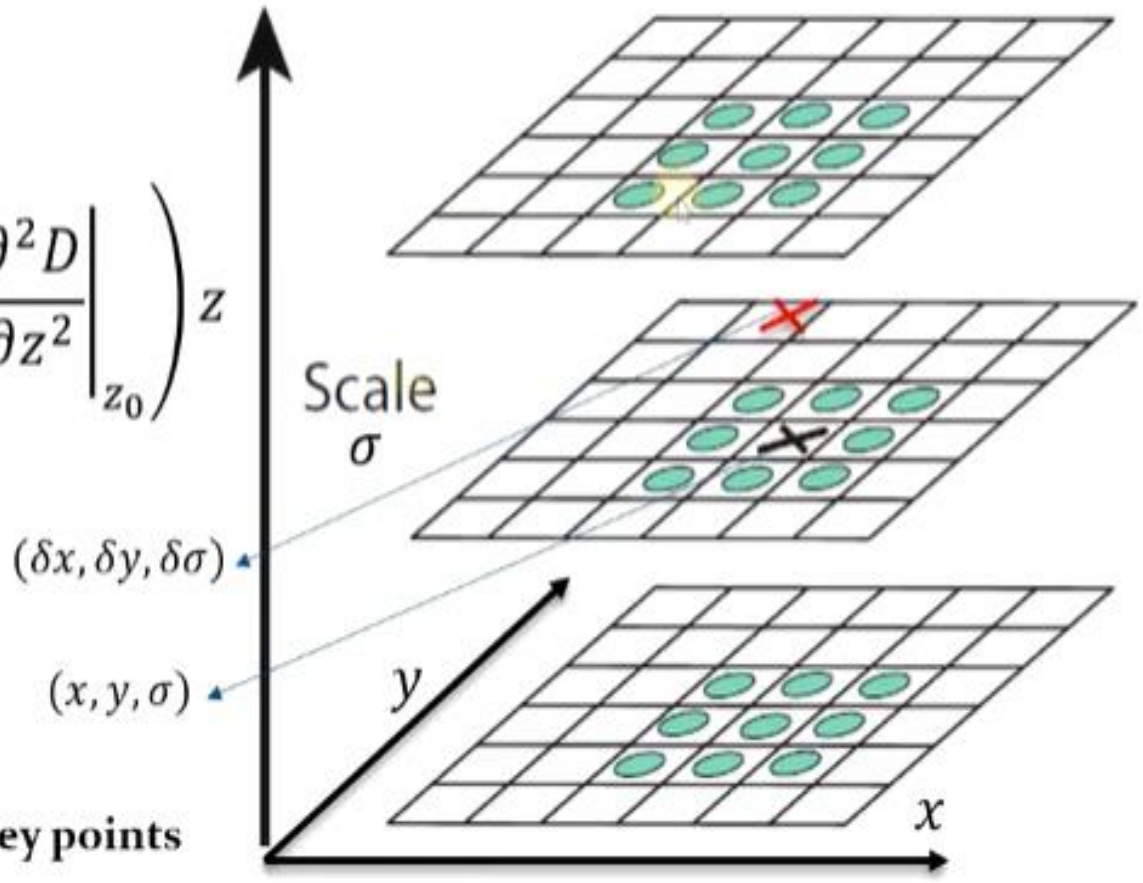
Interpolation using by fitting a Taylor expansion to fit a 3D quadratic surface (in x,y, and σ)

$$z_0 = [\, x, y, \sigma \,]^T \qquad z = [\, \delta x, \delta y, \delta \sigma \,]^T$$

$$D(z_0 + z) \approx D(Z_0) + \left(\frac{\partial D}{\partial z}\bigg|_{z_0} \right)^T z + \frac{1}{2} z^T \left(\frac{\partial^2 D}{\partial z^2}\bigg|_{z_0} \right) z$$

Finding the maxima or minima

$$\frac{\partial D(z_0 + z)}{\partial z} = 0$$



Distinctive Image Features from Scale-Invariant Key points
David G. Lowe

Feature Point Localization

$$D(z_0 + z) \approx D(Z_0) + \left(\frac{\partial D}{\partial z}\bigg|_{z_0}\right)^T z + \frac{1}{2} z^T \left(\frac{\partial^2 D}{\partial z^2}\bigg|_{z_0}\right) z$$

$$\frac{\partial D(z_0 + z)}{\partial z} = 0 + \left(\frac{\partial D}{\partial z}\bigg|_{z_0}\right) + \frac{1}{2} \times 2 \times \left(\frac{\partial^2 D}{\partial z^2}\bigg|_{z_0}\right) \hat{z} = 0$$

$$\left(\frac{\partial^2 D}{\partial z^2}\bigg|_{z_0}\right) \hat{z} = -\left(\frac{\partial D}{\partial z}\bigg|_{z_0}\right) \qquad \hat{z} = -\left(\frac{\partial^2 D}{\partial z^2}\bigg|_{z_0}\right)^{-1} \left(\frac{\partial D}{\partial z}\bigg|_{z_0}\right)$$

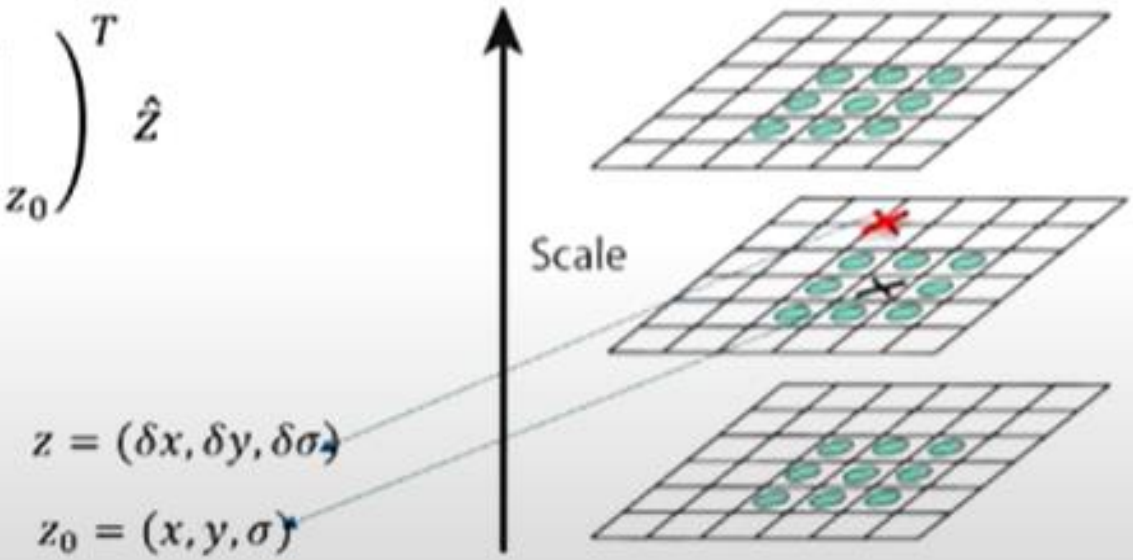
Feature Point Localization

$$D(z_0 + \hat{z}) \approx D(Z_0) + \left(\frac{\partial D}{\partial z}\Big|_{z_0}\right)^T \hat{z} - \frac{1}{2}\left(\frac{\partial D}{\partial z}\Big|_{z_0}\right)^T \left(\left(\frac{\partial^2 D}{\partial z^2}\Big|_{z_0}\right)^{-1}\right)^T \left(\frac{\partial^2 D}{\partial z^2}\Big|_{z_0}\right) \hat{z}$$

Hessian matrix

$$D(z_0 + \hat{z}) \approx D(Z_0) + \left(\frac{\partial D}{\partial z}\Big|_{z_0}\right)^T \hat{z} - \frac{1}{2}\left(\frac{\partial D}{\partial z}\Big|_{z_0}\right)^T \hat{z}$$

$$D(z_0 + \hat{z}) \approx D(Z_0) + \frac{1}{2}\left(\frac{\partial D}{\partial z}\Big|_{z_0}\right)^T \hat{z}$$



Removing points on the edges

The principal curvatures can be computed from a 2x2 Hessian matrix, H, computed at the location and scale of the key point

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

α and β are eigen values of the matrix

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r},$$

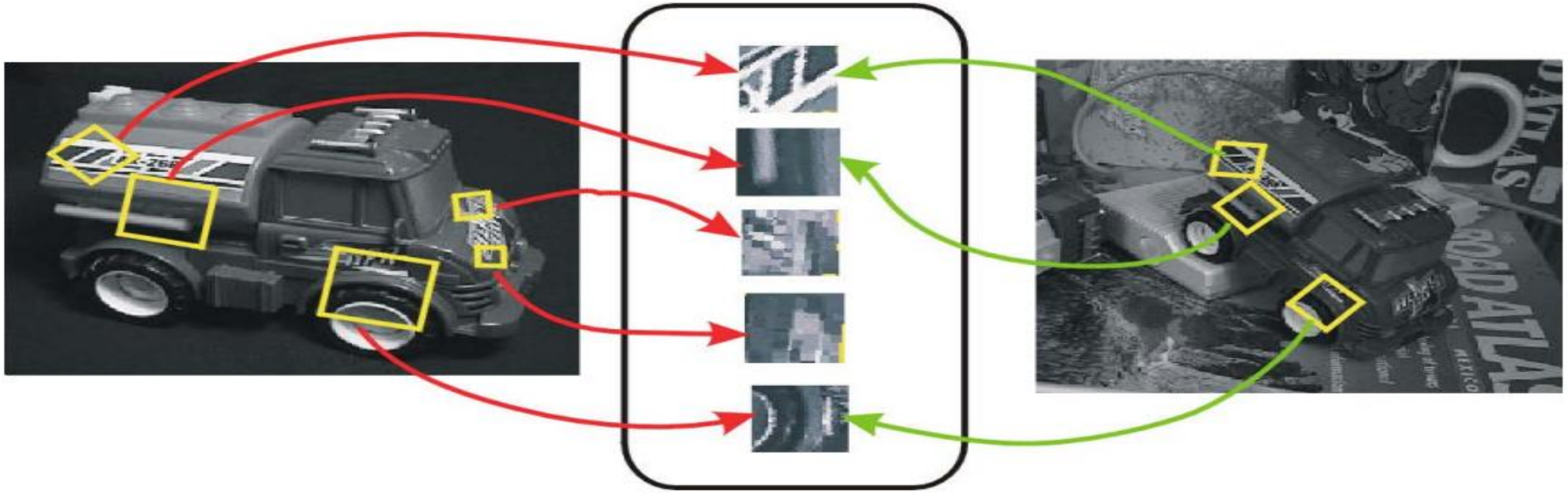
Keep only values that follow this condition

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} < \frac{(r+1)^2}{r}.$$

r=10

Idea of SIFT

- Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



SIFT Features

SIFT vector formation

- 4x4 array of gradient orientation histograms over 4x4 pixels
  - not really histogram, weighted by magnitude
- 8 orientations x 4x4 array = 128 dimensions
- Motivation: some sensitivity to spatial layout, but not too much.

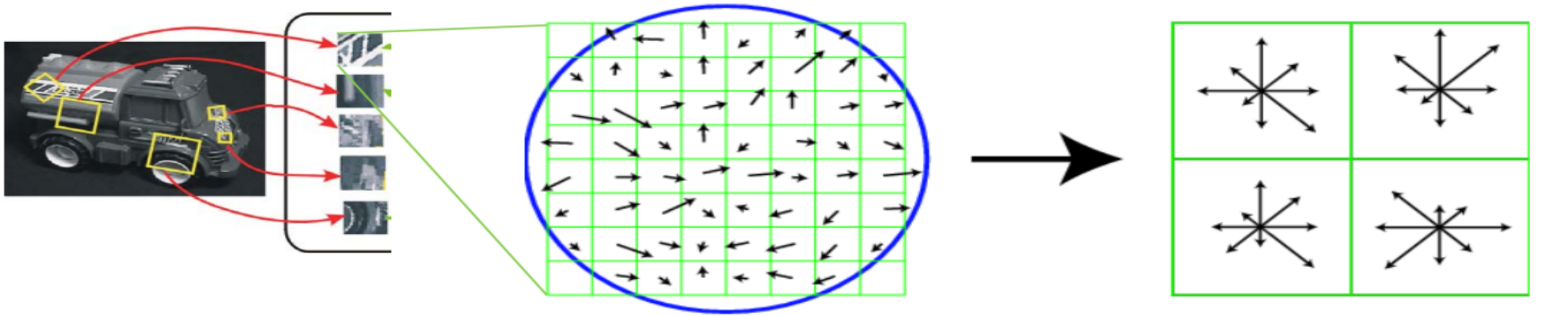


Image gradients

Keypoint descriptor

showing only 2x2 here but is 4x4

RANSAC – RANdom Sample Consensus

- RANdom SAmple Consensus
- Fischler & Bolles 1981
- Copes with a large proportion of outliers

RANSAC

Let's consider autonomous car navigation in cities.

Sensors are used to detect signals from the environment to identify moving and not moving objects such ground level, lanes, road edge, etc.

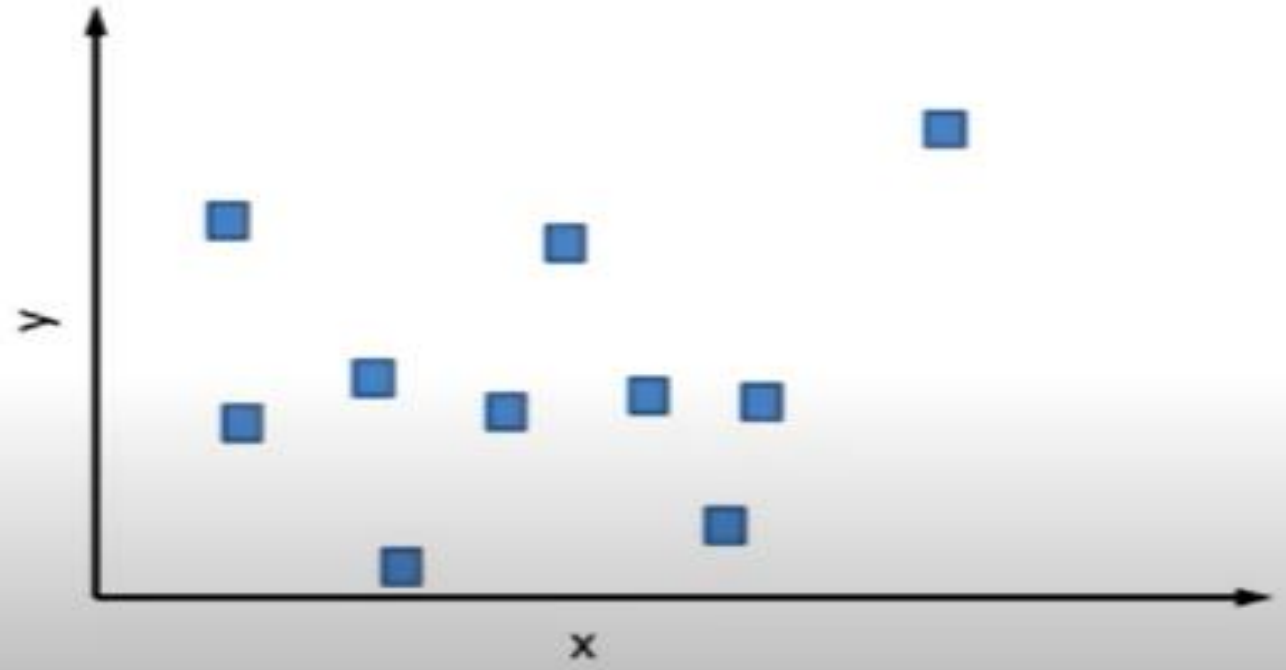
One of the common issues is the separation of the noisy data from the robust signal.

For instance, consider a wall near the road edge. Noise may shift the detected edge and car might crash into a building or pedestrian.

Least Squares Regression

Suppose we are given the data form sensors to detect the ground line.

Let's try a common line fitting technique – **Least Squares Regression**.



Least Squares Regression

Least squares method determines the best fit for a set of data points by minimizing the sum of the offsets or residuals of points from the plotted curve:

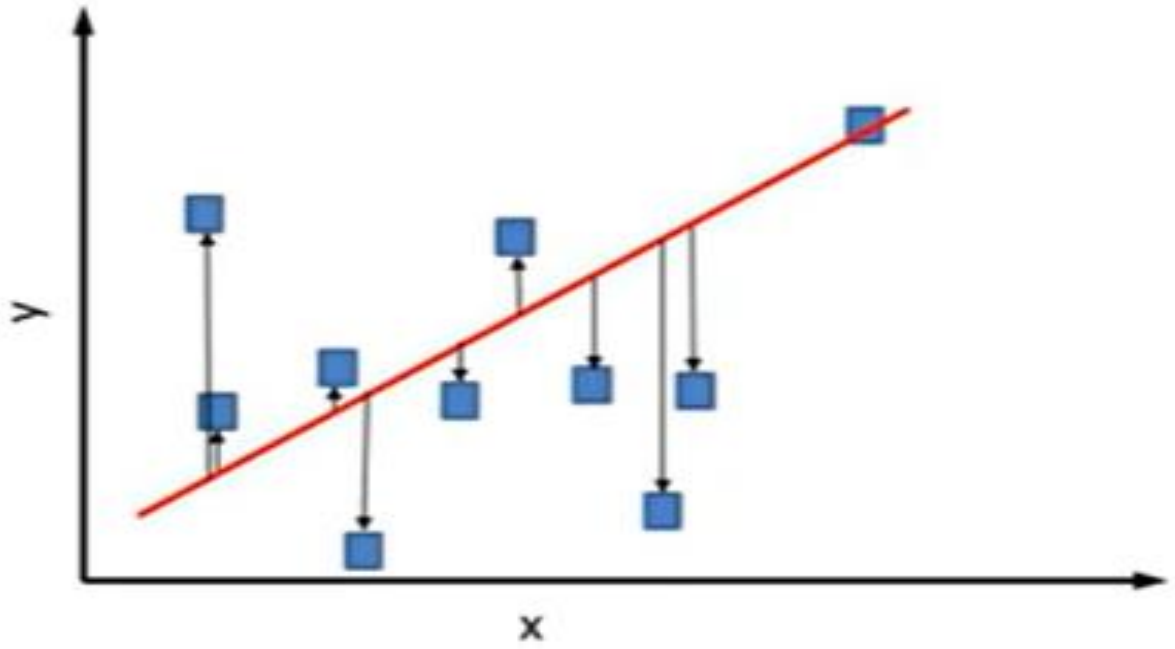
$$E = \sum_{i=1}^{n} (y_i - f(x_i, b_0, b_1, \dots, b_m))^2$$

We use the following function as the line equation:

$$f(x) = b_0 x + b_1$$

Least Squares Regression

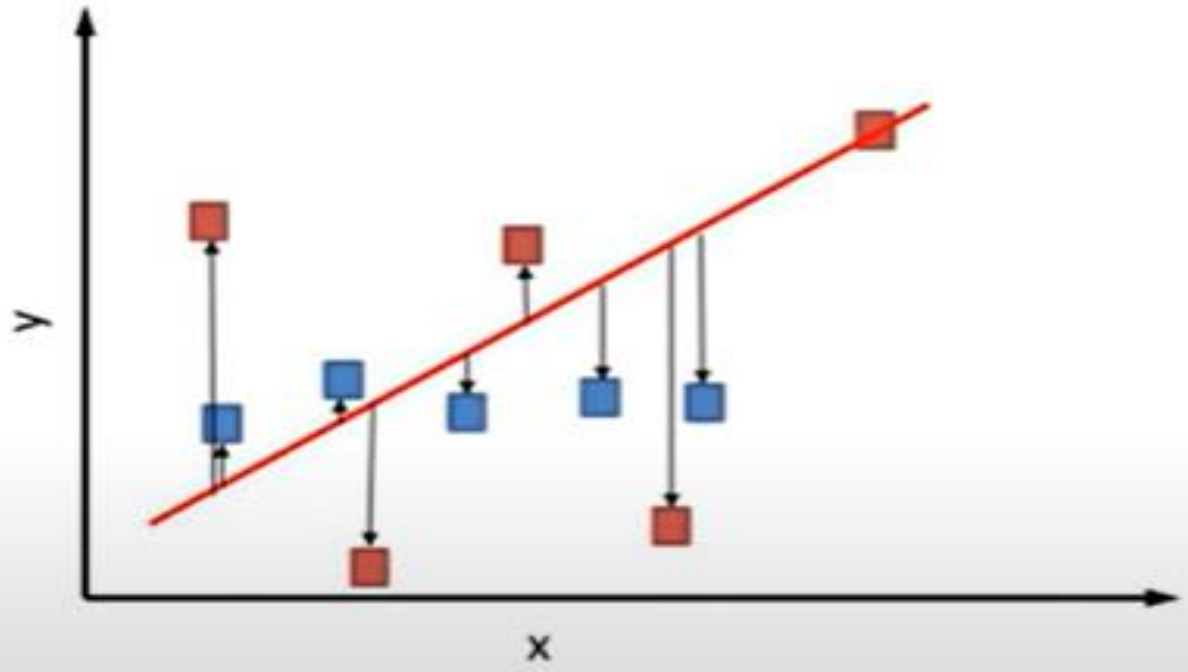
Least Squares Regression considers all data points and it is sensitive to outliers.



Least Squares Regression

Least Squares Regression considers all data points and it is sensitive to outliers.

Outliers are considered as noise that should be removed in order to make more reliable prediction.

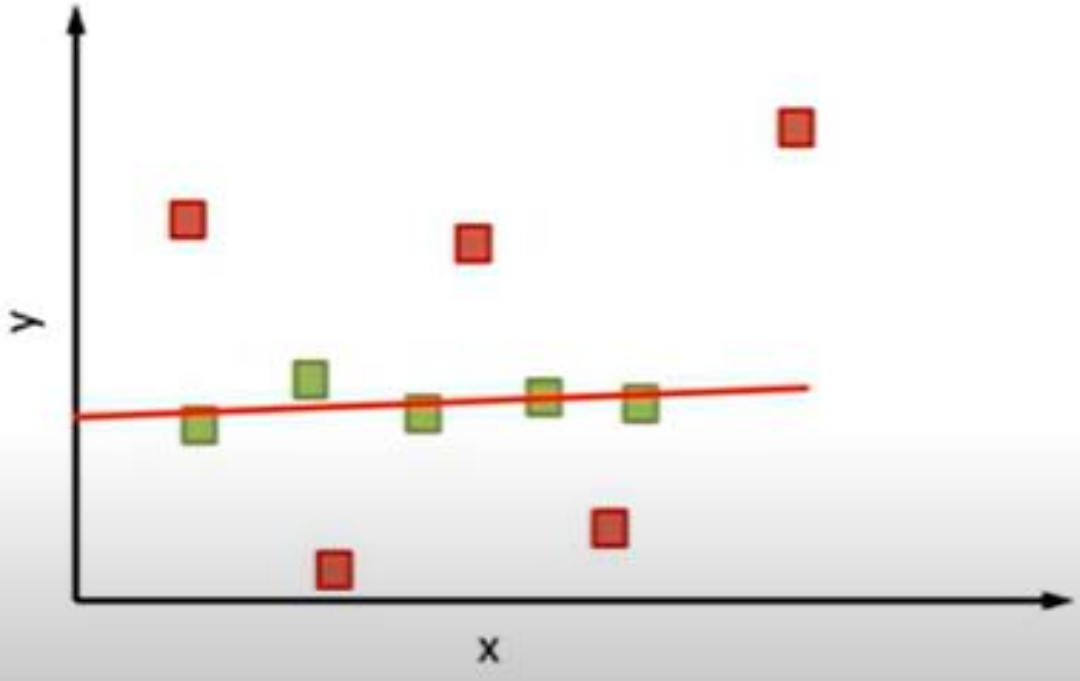


Outlier is an observation that deviates too much from others. The outliers are usually generated by a different mechanism from other observations.

Least Squares Regression

Least Squares Regression considers all data points and it is sensitive to outliers.

The trend for the fitted line should have been along the inliers (green data points).

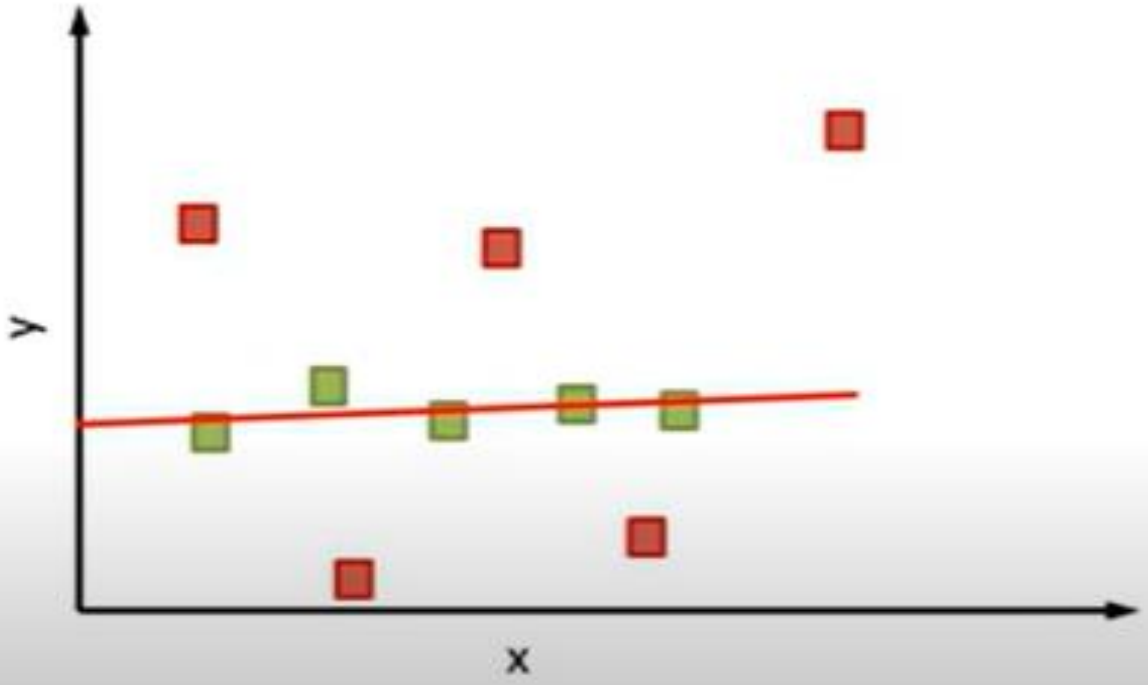


Inlier is an observation that is within the underlying probability distribution function and fits the predicted model well.

Least Squares Regression

Least Squares Regression considers all data points and it is sensitive to outliers.

The trend for the fitted line should have been along the inliers (green data points).



Inlier is an observation that is within the underlying probability distribution function and fits the predicted model well.

RANSAC

The main idea behind RANSAC is the **voting scheme**.

Main assumptions of RANSAC:

- Outliers or noisy data points do not vote consistently across models (few outliers).
- **Ratio of inliers to outliers is sufficient to predict the correct pattern.**

RANSAC Algorithm

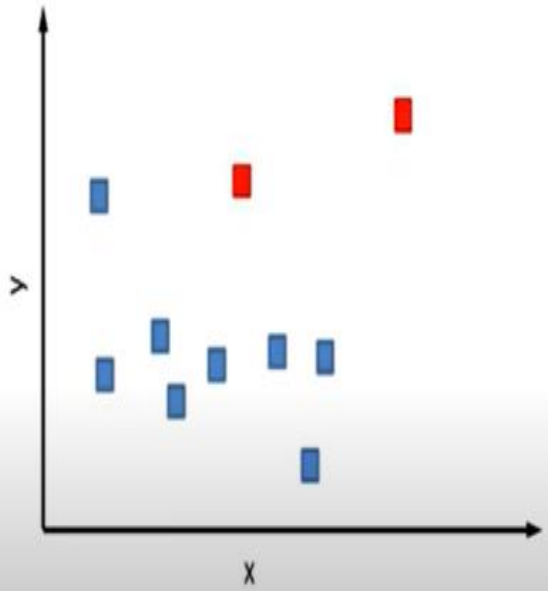
1. Select randomly M points from the dataset of points to determine the model parameters that fit the data.

2. Compute the parameters and get the model.

3. Check all the data points to find all the inliers to the model within the **error tolerance threshold** (δ)**.** If the inlier to total number ratio exceeds the predefined **minimum valid consensus threshold** (d)**,** correct the parameters with the identified inliers and stop.

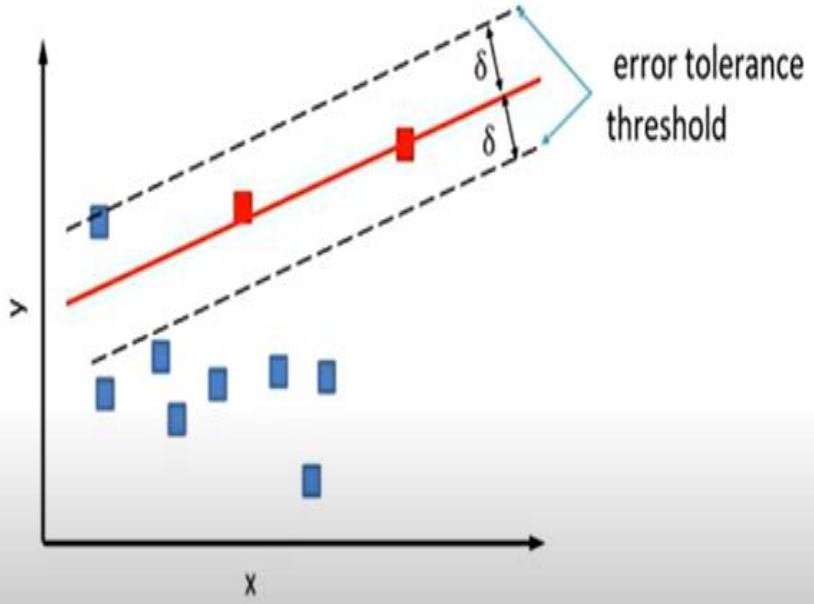
RANSAC

Because we use a linear function, we select randomly two points, i.e. **M = 2 for** Trial 1.



Trial 1. The ratio $\frac{inliers}{total} = \frac{3}{10} = 0.3$

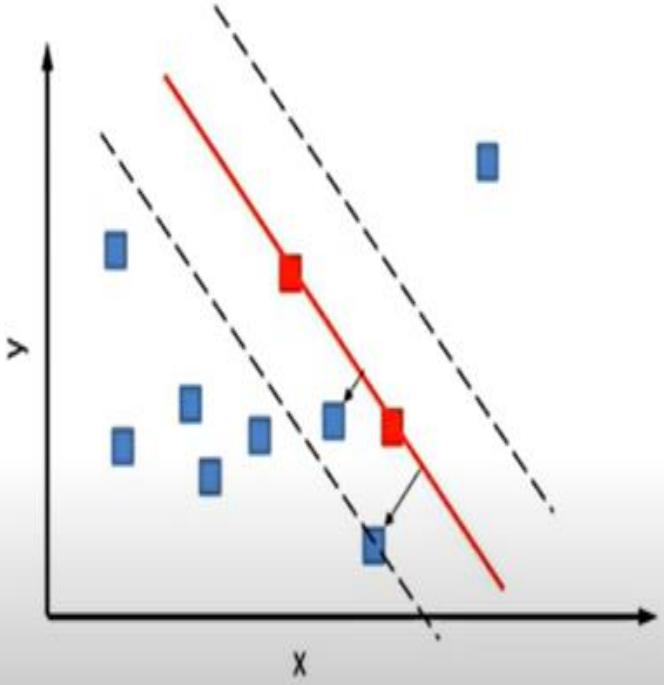
Not acceptable!



RANSAC

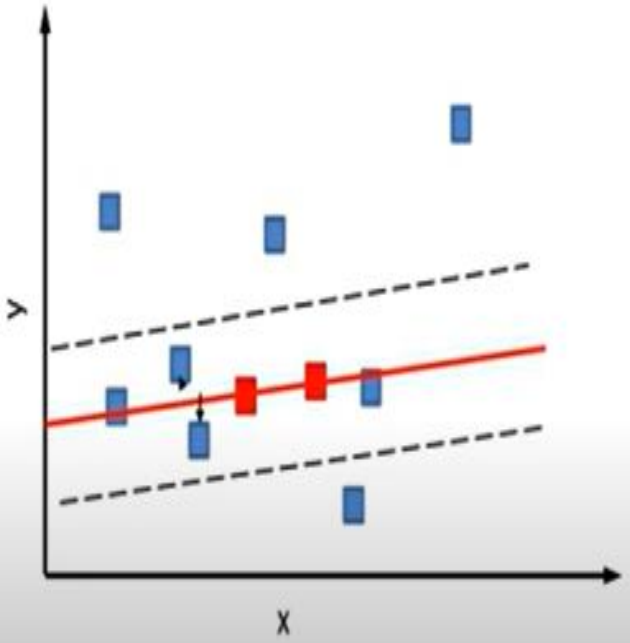
Trial 2. The ratio $\frac{inliers}{total} = \frac{4}{10} = 0.4$

Not acceptable!



Trial 3. The ratio $\frac{inliers}{total} = \frac{6}{10} = 0.6 > d$ **(minimum valid consensus threshold)**

Seems OK!



RANSAC

N can be set to a **large** number – at least one combination will consist of **inliers**!

Let k denote the probability that a randomly selected point is an inlier.

Then, $(1 - k)$ will be the probability of a random point to be an outlier.

The probability that in one selected sample set, all points will be inliers is

$$p = k^M$$

Then, the probability that in one picked set of samples, there will be an outlier becomes:

$$1 - p = (1 - k^M)$$

RANSAC

Probability for N iterations:

$$1 - p = (1 - k^M)^N \qquad (1)$$

Then, by solving (1), N can be derived as:

$$N = \frac{\log(1-p)}{\log(1-(1-k)^M)}$$

RANSAC

Number of iterations N required for the different outlier ratios:

M	Proportion of Outliers						
	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

RANSAC

Number of iterations N required for the different outlier ratios:

M	Proportion of Outliers						
	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

RANSAC

Studies show that proportion of outliers in the dataset should be less than **50%** for the RANSAC to work properly.

M	Proportion of Outliers						
	5%	10%	20%	25%	30%	40%	**50%**
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

RANSAC

Another critical parameter is the **error tolerance threshold**:

A suitable threshold value should be fine tuned.

If threshold is chosen too large, some random line considered as valid consensus will include some outliers.

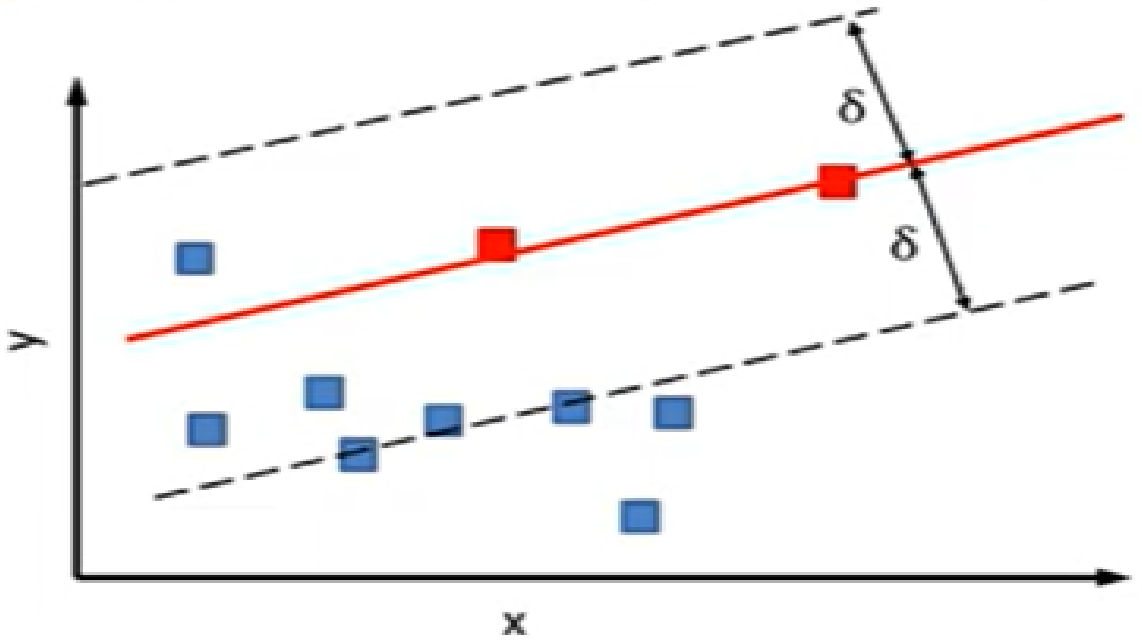
RANSAC

Another critical parameter is the **error tolerance threshold**:

A suitable threshold value should be fine tuned.

If threshold is chosen too large, some random line considered as valid consensus will include some outliers.

Consider trial 2 form the earlier example. The ratio $\frac{inliers}{total} = \frac{7}{10} = 0.7 \; - OK?$



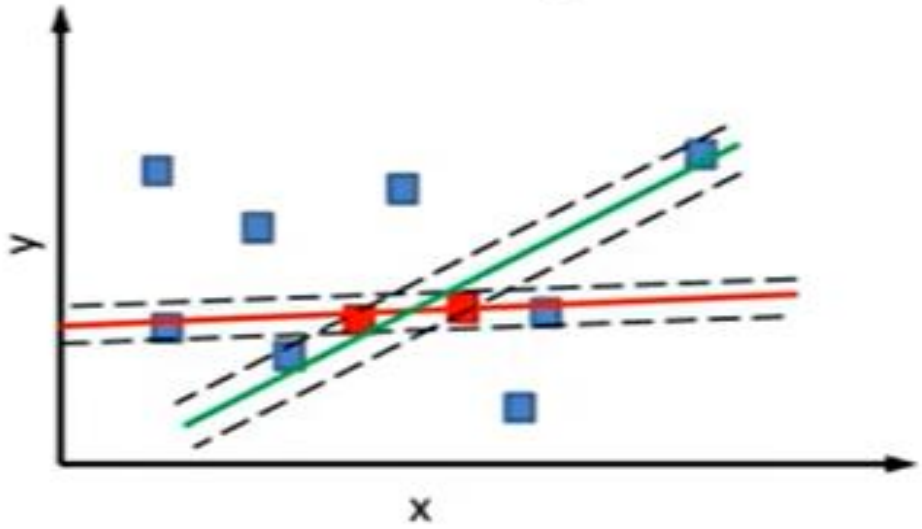
RANSAC

Another critical parameter is the **error tolerance threshold**:

A suitable threshold value should be fine tuned.

In the case of too small threshold, RANSAC will not reach a consensus that includes all inliers even if it choses two inliers for fitting the line.

Consider trial 2 form the example. The ratio $\frac{inliers}{total} = \frac{4}{10} = 0.4 - \boldsymbol{OK}?$



RANSAC

Advantages:

- RANSAC is easy to implement. Applicable to large datasets.
- It performs very well and deals with noisy data and outliers.

Disadvantages:

- Many parameters to tune. Tuning may take long.
- Requires reasonable inlier to outlier ratio. (should be greater than 50%)

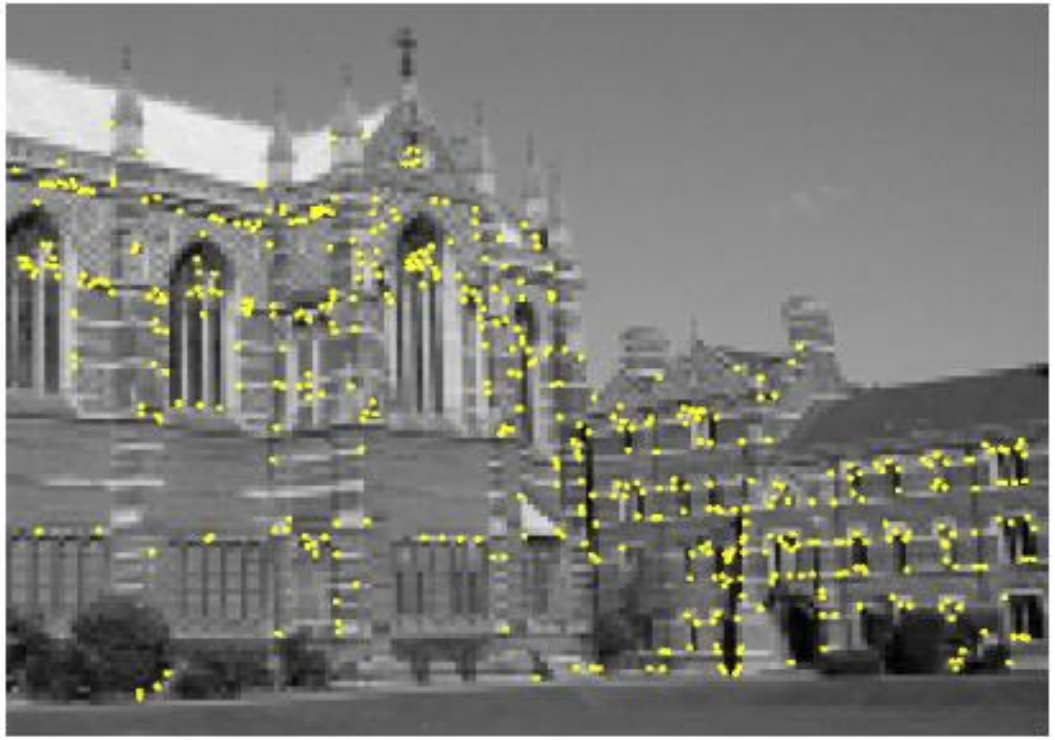
RANSAC

RANSAC can perform well in:

- Object feature translation
- Homographic transformations
- Detecting vanishing points
- Identify facial landmarks etc
- Identify similarities in images

RANSAC for recognition

RANSAC for fundamental matrix



RANSAC for fundamental matrix

Inliers (99)

Outliers (89)

RANSAC conclusions

The not-so-good…

- Computational time grows quickly with the number of model parameters
- *Sometimes problematic for approximate models*

Common applications

- Computing a homography (e.g., image stitching) or other image transform
- Estimating fundamental matrix (relating two views)
- *Pretty much every problem in robot vision*

RANSAC conclusions

The good…

- Simple and general
- Applicable to many different problems, often works well in practice
- *Robust to large numbers of outliers*
- Applicable for larger number of parameters than Hough transform
- Parameters are easier to choose than Hough transform