Edge Detection using Image Processing

AS AN APPLICATION OF LINEAR ALGEBRA

original

What is edge detection?

in digital images, which are just pictures made up of tiny dots called pixels, edges are places where the pixel values change a lot.

edge detection helps identify the boundaries or borders between different objects or regions in an image by looking for areas where the colours change a lot.

pixelated version

to perform edge detection, we're using :

Canny edge detection algorithm

Steps

- GAUSSIAN SMOOTHING / NOISE DETECTION
- GRADIENT CALCULATION
- NON - MAXIMUM SUPPRESSION
- DOUBLE THRESHOLDING
- EDGE TRACKING BY HYSTERESIS

1. Gaussian Smoothing / Noise Reduction

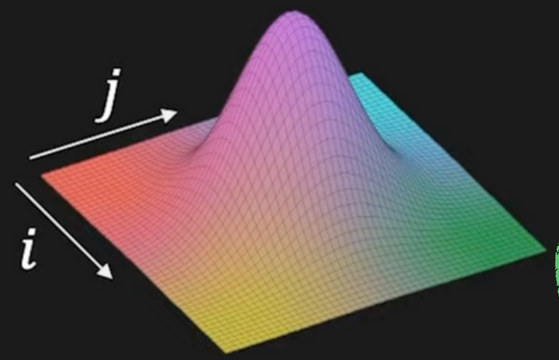
Mathematically, applying a Gaussian blur to an image is the same as convolving the image with a Gaussian function

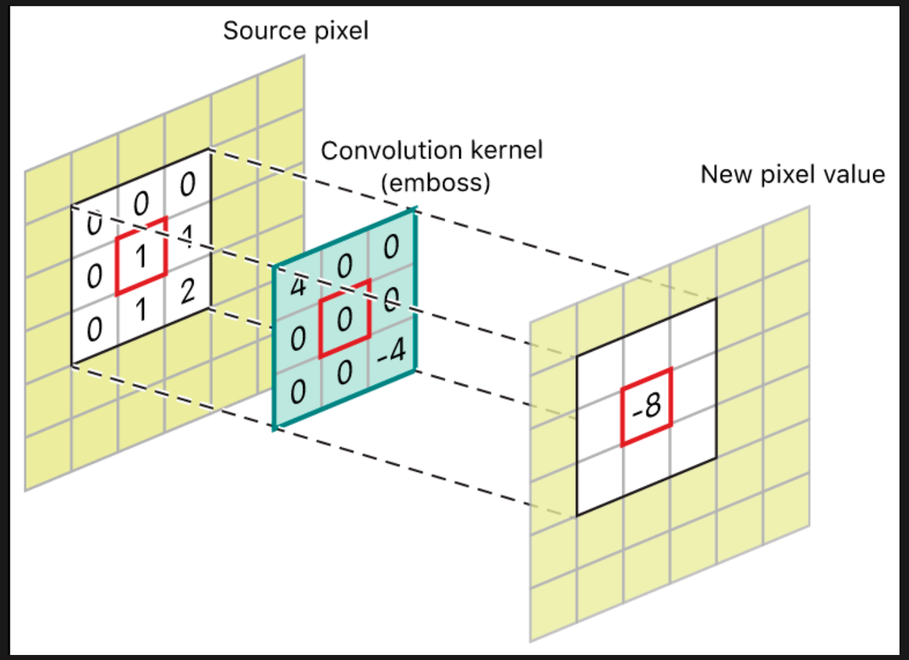
what is the gaussian function?

$$n_\sigma[i,j] = \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\left(\frac{i^2+j^2}{\sigma^2}\right)}$$

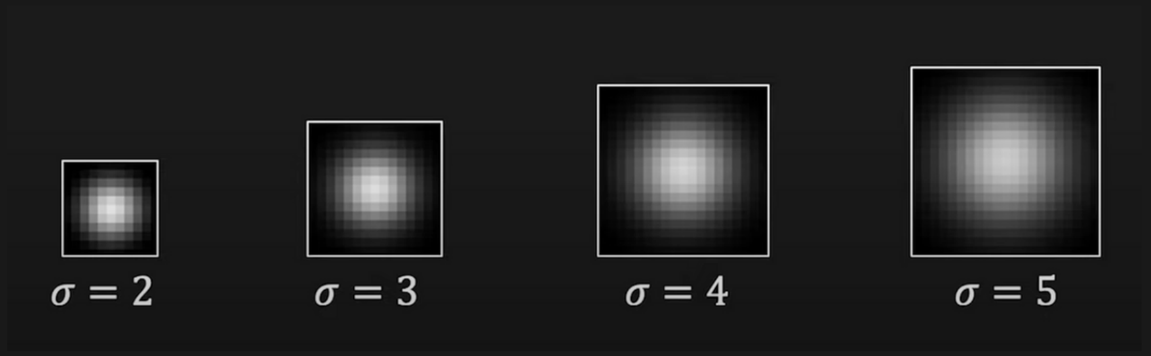
$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2}\right); 1 \le i, j \le (2k+1)$$

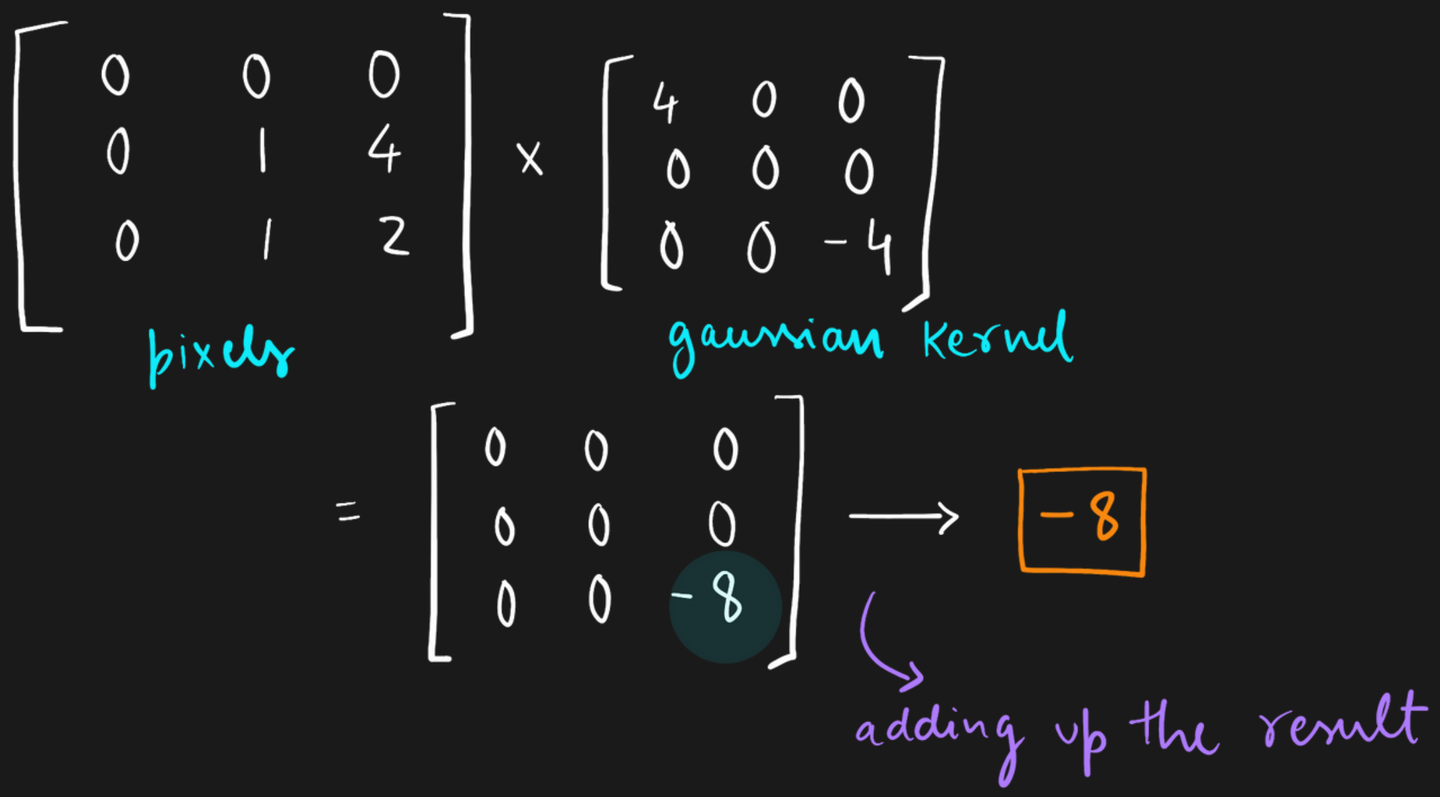
- nσ[i , j] is the value of the Gaussian function at a particular point [i , j] in the image with standard deviation σ
- σ (sigma) is a parameter called the standard deviation of the Gaussian function, which determines the width or spread of the curve. A larger σ value results in a wider curve and more smoothing





- The entries or values in the Gaussian kernel are calculated based on the Gaussian function formula.
- The Gaussian kernel is usually centered at the middle of the matrix or array, and the standard deviation (σ) of the Gaussian function determines the size or spread of the kernel.
- The larger the σ value, the wider the kernel.





- Repeat this process for every pixel in the image, applying the Gaussian kernel and updating the intensity values accordingly.
- The resulting image will be a smoothed or blurred version of the original image, with the degree of smoothing controlled by the standard deviation (σ) of the Gaussian function.

The result :

original

after applying gaussian blur

2.Gradient calculation

- Mathematically, the gradient of a two-variable function (here the image intensity function) at each image point is a 2D vector with the components given by the derivatives in the horizontal and vertical directions
- The most common way to approximate the image gradient is to convolve an image with a kernel, such as the Sobel operator or Prewitt operator.

What is a gradient function?

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

where,

- ∇f, f is a vector-valued function.

$$\frac{\partial f}{\partial x} \approx (f(i+1, j) - f(i-1, j))/2$$

$$\frac{\partial f}{\partial y} \approx (f(i, j+1) - f(i, j-1))/2$$

our matrix = $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ 1 & -1 & -1 \end{bmatrix}$ → after $\frac{\partial f}{\partial x}$ → $\begin{bmatrix} 2-0/2 & 1-1/2 & 1-2/2 \\ 0-1/2 & 0-0/2 & 0-0/2 \\ -1-1/2 & -1+1/2 & -1+0/2 \end{bmatrix}$

↓

$$\begin{bmatrix} 1/2 & 0 & -1/2 \\ -1/2 & 0 & 1/2 \\ -1 & 0 & 1 \end{bmatrix}$$

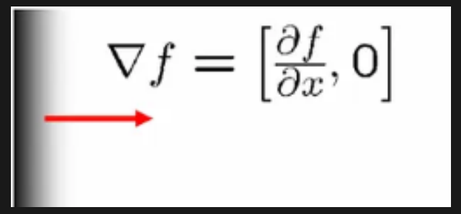
III ly,

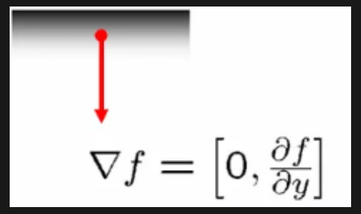
$$\frac{\partial f}{\partial y} = \begin{bmatrix} -1/2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1/2 & -1/2 \end{bmatrix}$$

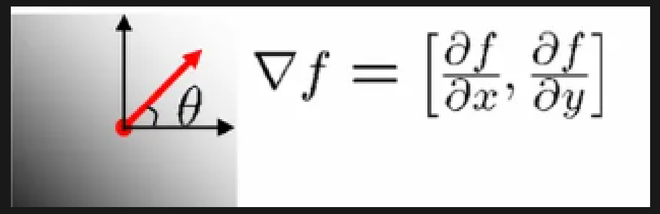
The Gradient matrix represents the strength of the edges at each pixel location, with higher values indicating stronger edges.

2.Gradient calculation

The gradient points in the x, y and both the direction respectively w.r.t intensity change:







The gradient direction(orientation of edge normal) is given by:

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

NOTE:

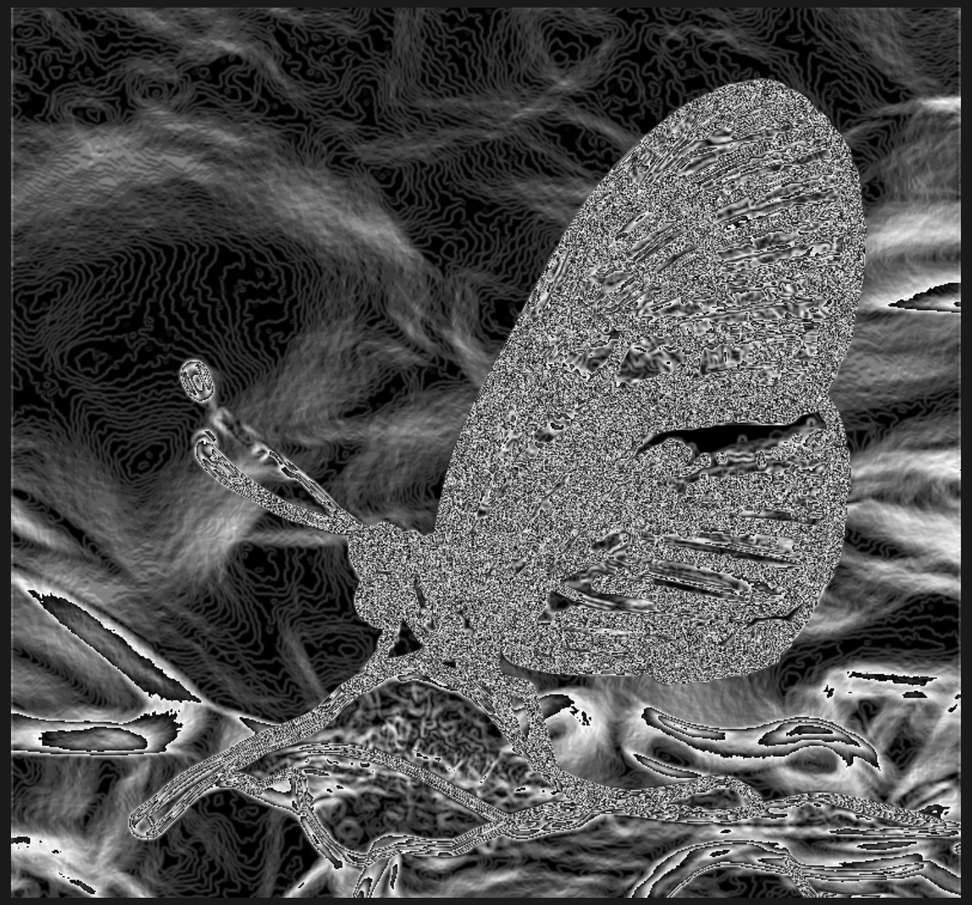
Gradient points in the direction of most rapid increase in intensity (theta)

The edge strength is given by the gradient magnitude:

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

2.Gradient calculation

original



after gradient calculation

3. Non-maximum suppression

refines the edges detected by gradient calculation

we get theta from gradient calculation, which should be quantised to the nearest multiple of 45 degrees

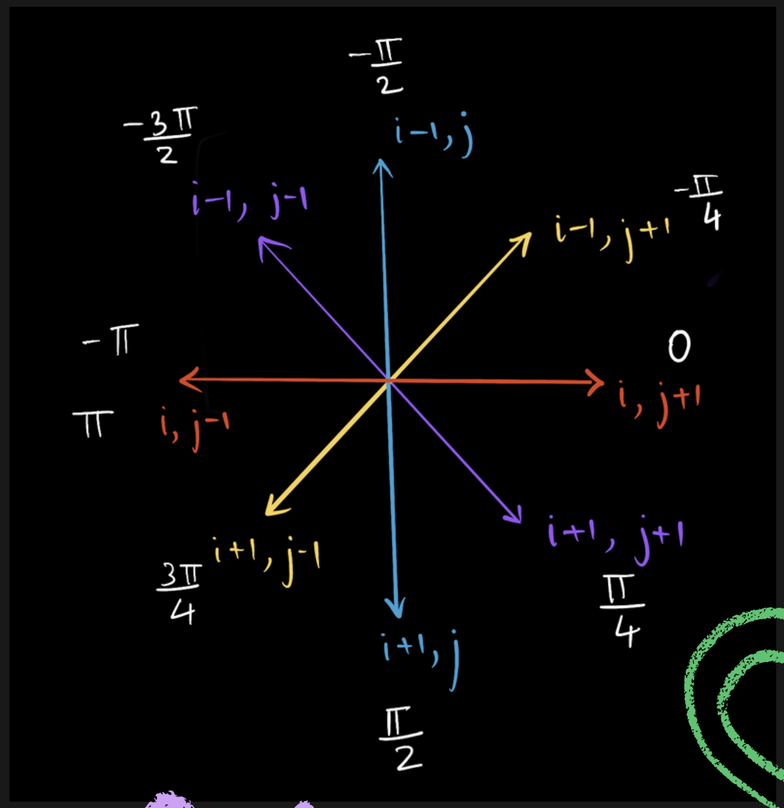
63.43	63.43	90.00
71.57	0.00	180.00
-116.57	-135.00	-135.00

quantised θ =

45	45	90
90	0	180
-135	-135	-135

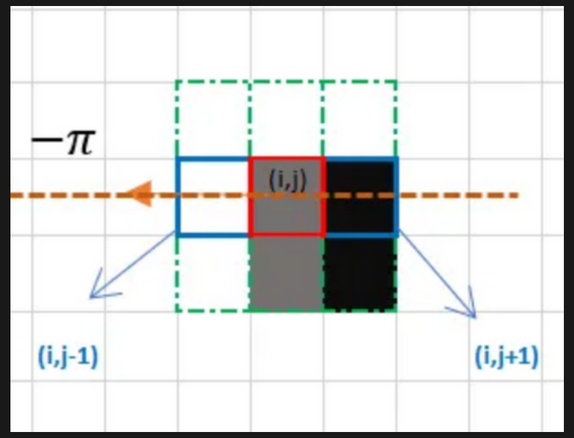
3 matrices are considered here:

- the pixel matrix after gradient calculation
- the angle matrix
- the output matrix after suppression



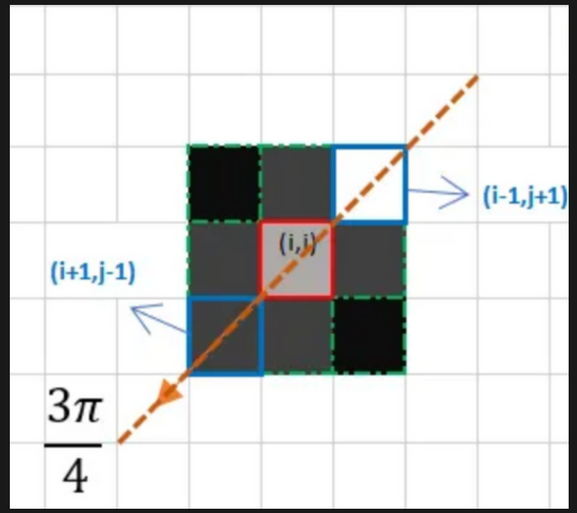
3. Non-maximum suppression

(directions are considered for each pixel using the quantised theta matrix)



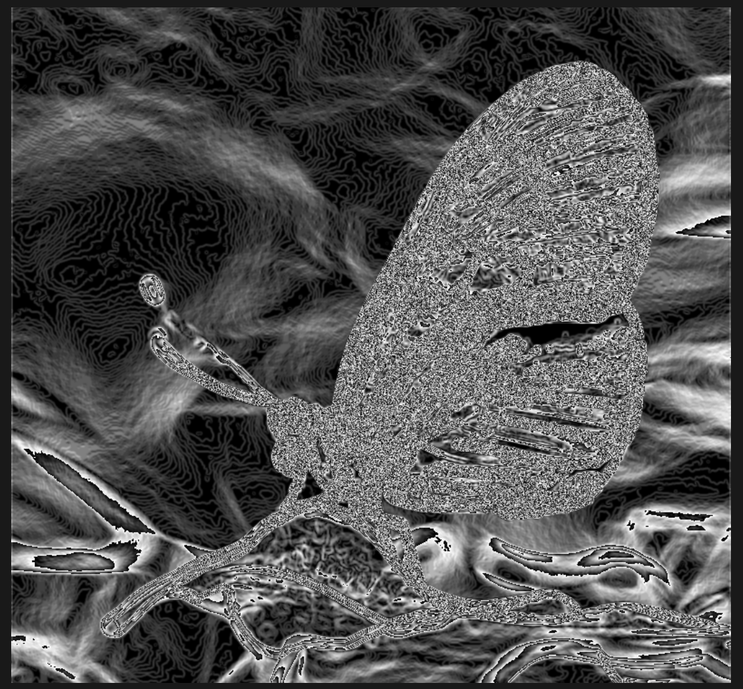
pixel at (i,j-1) has more intensity(255) , hence in the output matrix value of (i,j) is suppressed. it is made 0 in output matrix.

the most intense pixel in this direction is the pixel (i-1, j+1).

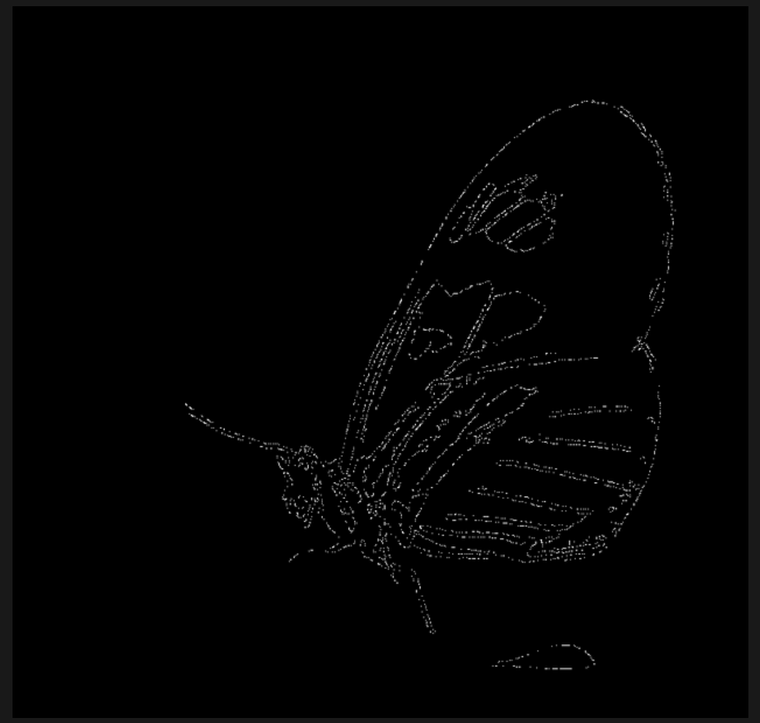


The result:

original



gradient calculation

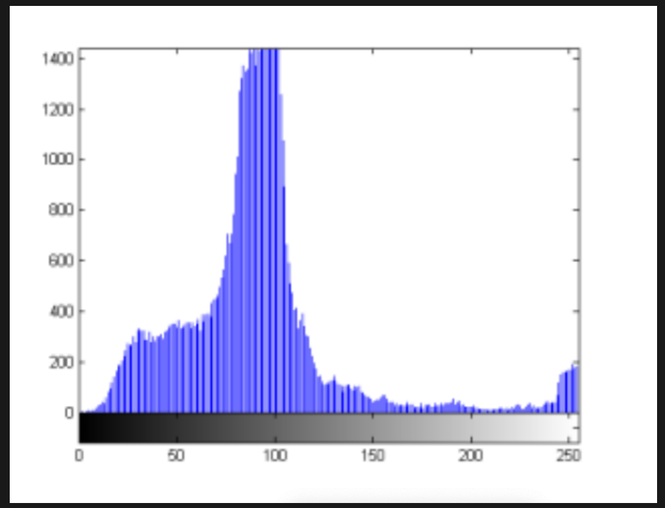


non maximum suppression

non-maximum suppression helps to enhance the accuracy and thinness of the detected edges in the final edge map

4. Double Thresholding

The double threshold step aims at identifying 3 kinds of pixels: strong, weak, and non-relevant



1. plot a histogram of the output matrix obtained from non maximum suppression
2. apply a gaussian filter
3. The threshold values are selected as a percentage of the maximum value in the smoothed histogram

how are pixels classified here?

pixel > high threshold

gives strong pixels

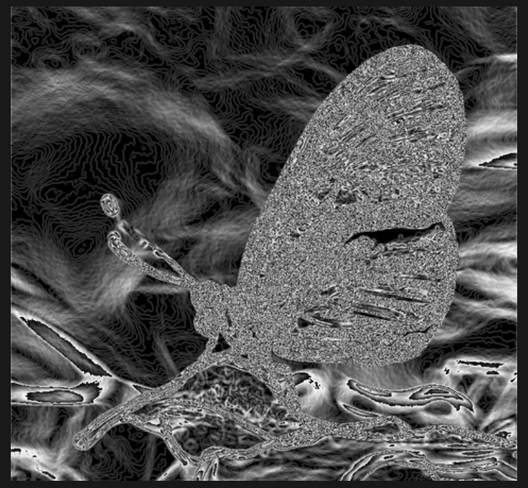
low threshold < pixel < high threshold

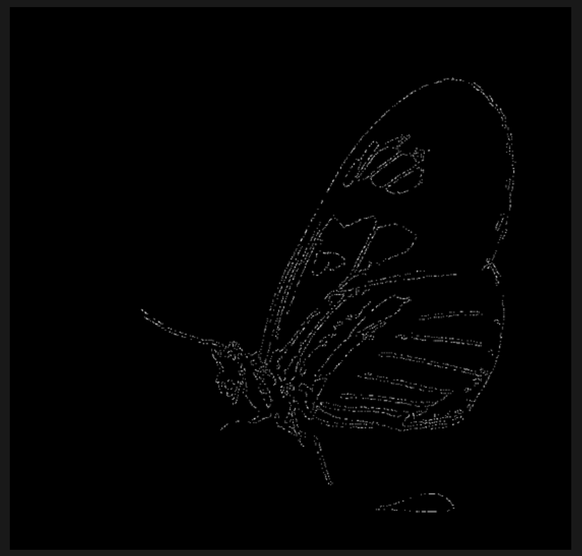
gives weak pixels

pixel < low threshold

gives non relevant pixels

The result:

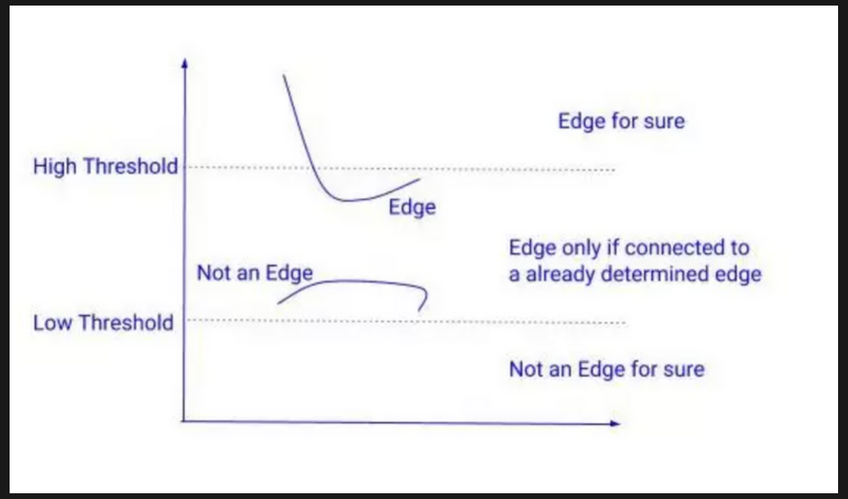




after double thresholding

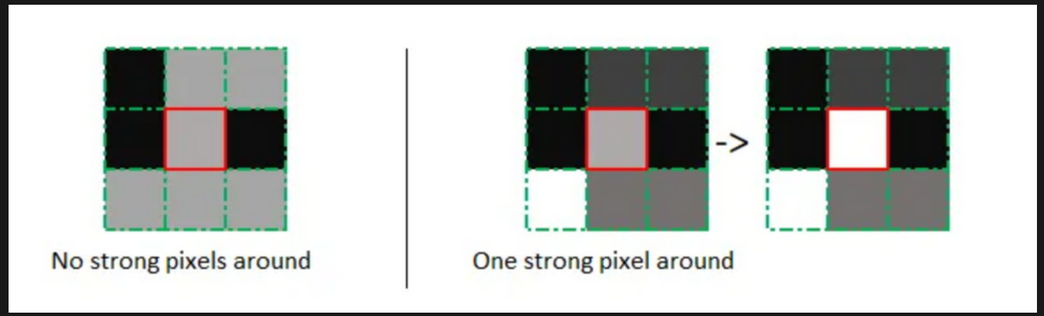
carried out with the assumption that the range of intensity levels covered by objects of interest is different from the background

5. Edge tracking by hysteresis



Strong edges are interpreted as “certain edges”,Weak edges are included if and only if they are connected to strong edges.

the hysteresis consists of transforming weak pixels into strong ones, if at least one of the pixels around the one being processed is a strong one



5. Edge tracking by hysteresis

original

Strong + connected weak edges

gap is not included

Strong edges only

Weak edges

courtesy of G. Loy

The result:

original

after edge hysteresis

Code and its Implementation

1. Gaussian kernel function:

```
def gaussian_kernel(self, size, sigma=1):
    size = int(size) // 2
    x, y = np.mgrid[-size:size+1, -size:size+1]
    normal = 1 / (2.0 * np.pi * sigma**2)
    g =  np.exp(-((x**2 + y**2) / (2.0*sigma**2))) * normal
    return g
```

2. Sobel Filter function:

```
def sobel_filters(self, img):
    Kx = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]], np.float32)
    Ky = np.array([[1, 2, 1], [0, 0, 0], [-1, -2, -1]], np.float32)

    Ix = ndimage.filters.convolve(img, Kx)
    Iy = ndimage.filters.convolve(img, Ky)

    G = np.hypot(Ix, Iy)
    G = G / G.max() * 255
    theta = np.arctan2(Iy, Ix)
    return (G, theta)
```

3. Non-Maximum Suppression function:

```
def non_max_suppression(self, img, D):
    M, N = img.shape
    Z = np.zeros((M,N), dtype=np.int32)
    angle = D * 180. / np.pi
    angle[angle < 0] += 180


    for i in range(1,M-1):
        for j in range(1,N-1):
            try:
                q = 255
                r = 255

               #angle 0
                if (0 <= angle[i,j] < 22.5) or (157.5 <= angle[i,j] <= 180):
                    q = img[i, j+1]
                    r = img[i, j-1]
                #angle 45
                elif (22.5 <= angle[i,j] < 67.5):
                    q = img[i+1, j-1]
                    r = img[i-1, j+1]
                #angle 90
                elif (67.5 <= angle[i,j] < 112.5):
                    q = img[i+1, j]
                    r = img[i-1, j]
                #angle 135
                elif (112.5 <= angle[i,j] < 157.5):
                    q = img[i-1, j-1]
                    r = img[i+1, j+1]

                if (img[i,j] >= q) and (img[i,j] >= r):
                    Z[i,j] = img[i,j]
                else:
                    Z[i,j] = 0


            except IndexError as e:
                pass

    return Z
```

4. Threshold function:

```
def threshold(self, img):

    highThreshold = img.max() * self.highThreshold;
    lowThreshold = highThreshold * self.lowThreshold;

    M, N = img.shape
    res = np.zeros((M,N), dtype=np.int32)

    weak = np.int32(self.weak_pixel)
    strong = np.int32(self.strong_pixel)

    strong_i, strong_j = np.where(img >= highThreshold)
    zeros_i, zeros_j = np.where(img < lowThreshold)

    weak_i, weak_j = np.where((img <= highThreshold) & (img >= lowThreshold))

    res[strong_i, strong_j] = strong
    res[weak_i, weak_j] = weak

    return (res)
```

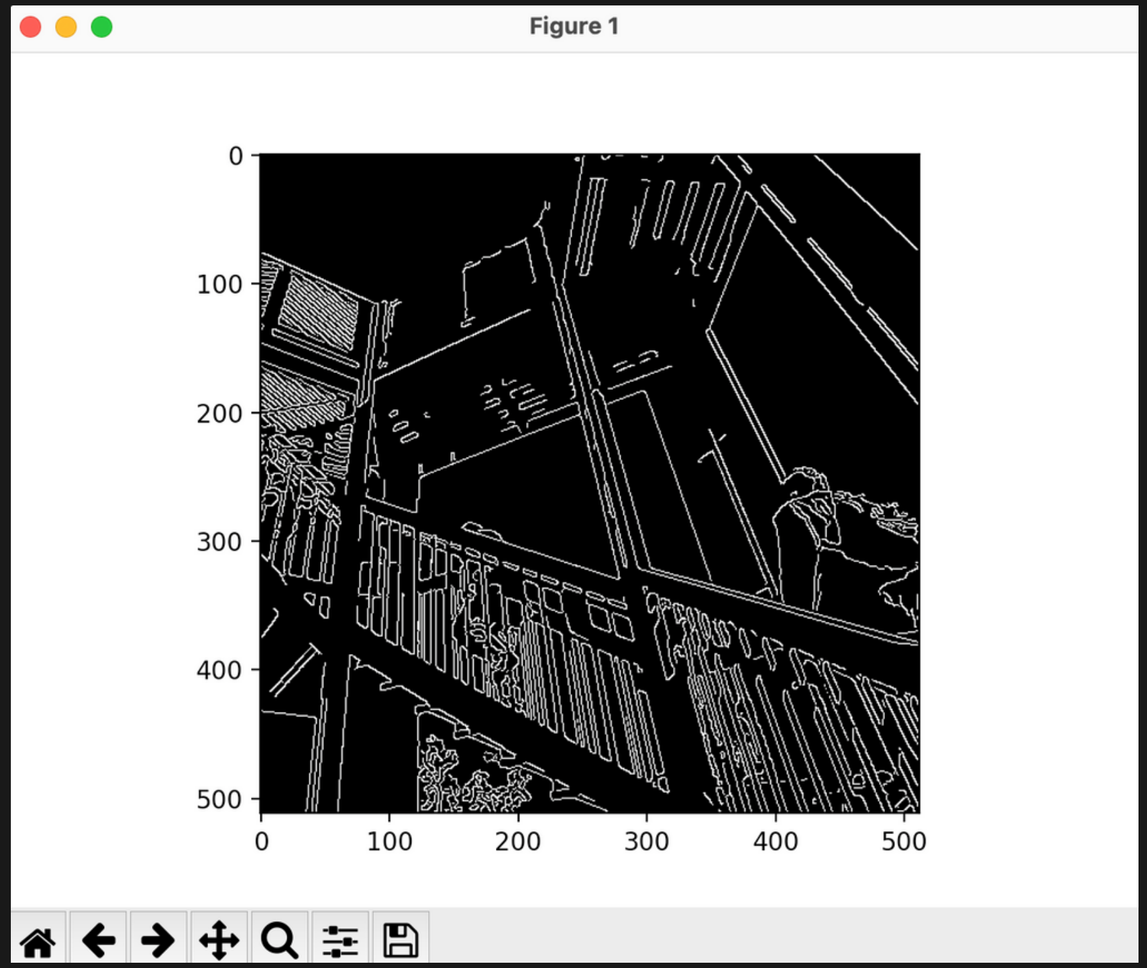
5. Hysteresis function:

```
def hysteresis(self, img):

    M, N = img.shape
    weak = self.weak_pixel
    strong = self.strong_pixel

    for i in range(1, M-1):
        for j in range(1, N-1):
            if (img[i,j] == weak):
                try:
                    if ((img[i+1, j-1] == strong) or (img[i+1, j] == strong) or (img[i+1, j+1] == strong)
                        or (img[i, j-1] == strong) or (img[i, j+1] == strong)
                        or (img[i-1, j-1] == strong) or (img[i-1, j] == strong) or (img[i-1, j+1] == strong)):
                        img[i, j] = strong
                    else:
                        img[i, j] = 0
                except IndexError as e:
                    pass

    return img
```

Calling the functions:

```
def detect(self):
    imgs_final = []
    for i, img in enumerate(self.imgs):
        self.img_smoothed = convolve(img, self.gaussian_kernel(self.kernel_size, self.sigma))
        self.gradientMat, self.thetaMat = self.sobel_filters(self.img_smoothed)
        self.nonMaxImg = self.non_max_suppression(self.gradientMat, self.thetaMat)
        self.thresholdImg = self.threshold(self.nonMaxImg)
        img_final = self.hysteresis(self.thresholdImg)
        self.imgs_final.append(img_final)

    return self.imgs_final
```

Output:



Real Life Applications of Edge Detection

FINGER PRINT RECOGNITION

SATELLITE IMAGING

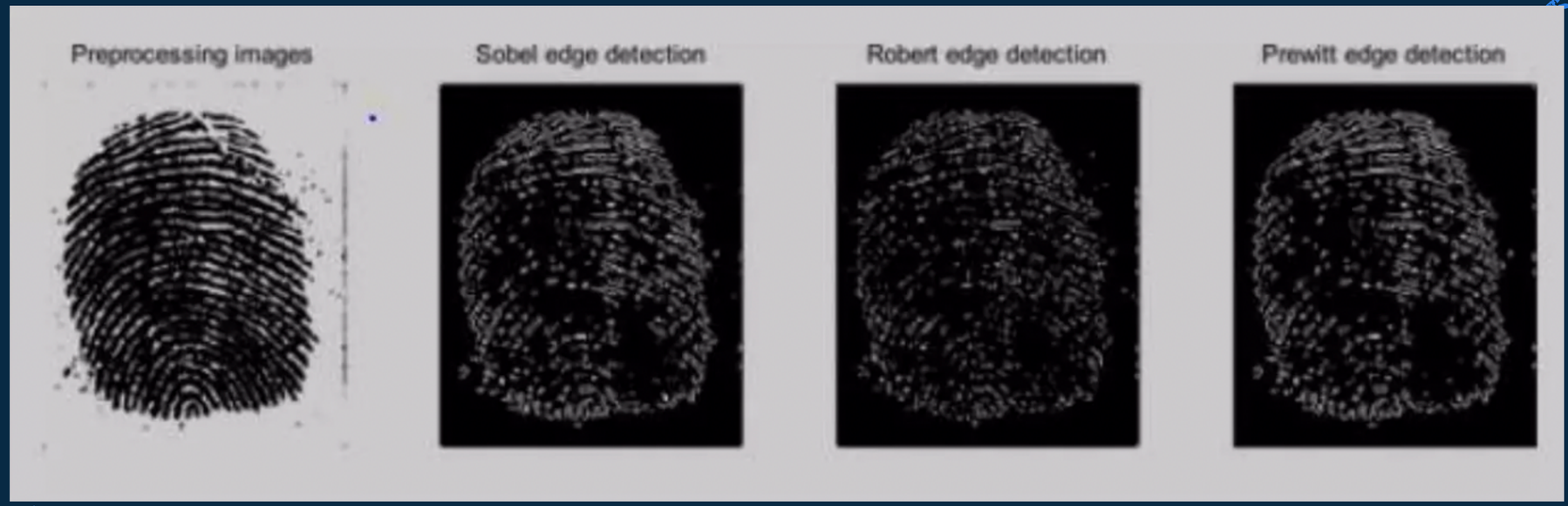
ROBOTICS VISION

MEDICAL SCANS

and many more...

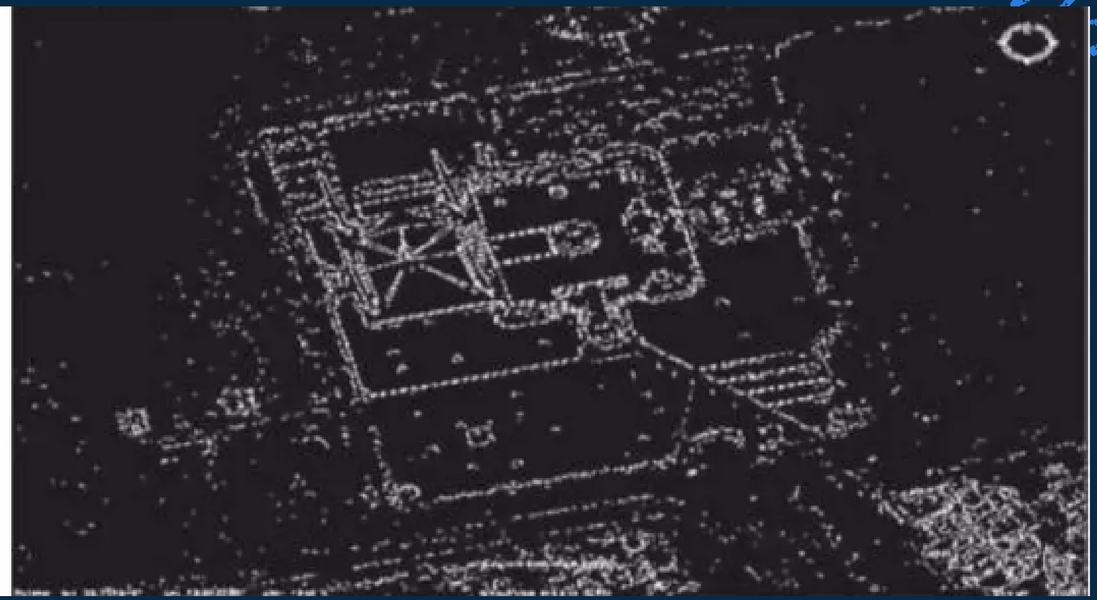
1. Finger Print Recognition

- At present, finger print recognition has been used widely, such as in mobile phones.
- Edge detection techniques enhance the quality of image and cause the improvement in the image recognition.



2. Satellite Imaging

- The edge map generated by bilateral filtering based edge contain more accurate and well localized edge map.
- It also suppresses noise and produces more realistic edge map.



3. Robotics Vision

- The present and near future main application areas of edge detection are robotics vision, like in self-driving vehicles.

4. Medical Scans

- Another important area being developed involving edge detection is finding "pathalogical" objects in medical images such as tumors.
- Medical image edge detection is an important method in the recognition of the human organs and it is an important pre-processing step in 3D reconstruction such as the reconstruction of brain images,

More Applications

...

- PORTRAIT MODE AND DEPTH EFFECTS IN CAMERAS
- IMAGE EDITING TO REMOVE BACKGROUND OBJECTS
- AUTONOMOUS VEHICLES
- QUALITY CONTROL TO DETECT DEFECTS AND FLAWS
- SURVEILLANCE TO DETECT MOVING OBJECTS

Advantages

- Good Accuracy
- Low error rate
- Handles noise well
- Single edge detection
- Tunable parameters

Disadvantages

- Computationally expensive
- Not suitable for all images
- Missing low-contrast edges
- Sensitivity to lighting changes

THANK YOU!

Presented by:

Dhiruram B

Chandana S M

Deeksha K