```
pip install opencv-contrib-python

pip install numpy

pip install opencv-python numpy matplotlib

pip install numpy

pip install "numpy<2.0"

pip install matplotlib

!pip install matplotlib
```

```
Requirement already satisfied: opencv-contrib-python in c:\users\achal\anaconda3\lib\site-packages (4.11.0.86)
Requirement already satisfied: numpy>=1.21.2 in c:\users\achal\anaconda3\lib\site-packages (from opencv-contrib-python) (1.26.4)
Note: you may need to restart the kernel to use updated packages.
```

geometreic operations- tralation

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt


img = cv2.imread(r"C:\Users\my.ac.p2mca24004.AMRITAMYSORE\Downloads\ct_healthy (10).jpg")
img = cv2.resize(img, (100, 100))

tx, ty = 30, 20
M = np.float32([[1, 0, tx],
                [0, 1, ty]])

res = cv2.warpAffine(img, M, (img.shape[1], img.shape[0]))

img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
res_rgb = cv2.cvtColor(res, cv2.COLOR_BGR2RGB)

fig, ax = plt.subplots(1, 2, figsize=(6, 3))
ax[0].imshow(img_rgb)
ax[0].set_title('Original Image')
ax[0].axis('off')

ax[1].imshow(res_rgb)
ax[1].set_title('Translated Image')
ax[1].axis('off')

plt.tight_layout()
plt.show()
```

geometreic operations- scaling

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread(r"C:\Users\my.ac.p2mca24004.AMRITAMYSORE\Downloads\ct_healthy (10).jpg")
img = cv2.resize(img, (100, 100))

scale_x, scale_y = 2.55, 2.55

scaled_img = cv2.resize(img, None, fx=scale_x, fy=scale_y, interpolation=cv2.INTER_LINEAR)

img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
scaled_rgb = cv2.cvtColor(scaled_img, cv2.COLOR_BGR2RGB)

fig, ax = plt.subplots(1, 2, figsize=(6, 3))
ax[0].imshow(img_rgb)
ax[0].set_title('Original Image')
ax[0].axis('off')

ax[1].imshow(scaled_rgb)
ax[1].set_title('Scaled Image (2.x)')
ax[1].axis('off')

plt.tight_layout()
plt.show()
```

geometreic operations- Rotation

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread(r"C:\Users\my.ac.p2mca24004.AMRITAMYSORE\Downloads\ct_healthy (10).jpg")
img = cv2.resize(img, (100, 100))

angle = 45
scale = 1.0

(h, w) = img.shape[:2]
center = (w // 2, h // 2)

M = cv2.getRotationMatrix2D(center, angle, scale)
rotated_img = cv2.warpAffine(img, M, (w, h))

img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
rotated_rgb = cv2.cvtColor(rotated_img, cv2.COLOR_BGR2RGB)

fig, ax = plt.subplots(1, 2, figsize=(6, 3))
ax[0].imshow(img_rgb)
ax[0].set_title('Original Image')
ax[0].axis('off')

ax[1].imshow(rotated_rgb)
ax[1].set_title(f'Rotated Image ({angle}°)')
ax[1].axis('off')

plt.tight_layout()
plt.show()
```

geometreic operations-Flipping

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread(r"C:\Users\my.ac.p2mca24004.AMRITAMYSORE\Downloads\ct_healthy (10).jpg")
img = cv2.resize(img, (100, 100))

flip_code = 1
flipped_img = cv2.flip(img, flip_code)

img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
flipped_rgb = cv2.cvtColor(flipped_img, cv2.COLOR_BGR2RGB)

fig, ax = plt.subplots(1, 2, figsize=(6, 3))
ax[0].imshow(img_rgb)
ax[0].set_title('Original Image')
ax[0].axis('off')

flip_label = {0: 'Vertical Flip', 1: 'Horizontal Flip', -1: 'Both Flipped'}[flip_code]
ax[1].imshow(flipped_rgb)
ax[1].set_title(f'{flip_label}')
ax[1].axis('off')

plt.tight_layout()
plt.show()
```

geometreic operations- sheared

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

image = cv2.imread(r"C:\Users\my.ac.p2mca24004.AMRITAMYSORE\Downloads\tumor.jpg")
image = cv2.resize(image, (300, 300))

height, width = image.shape[:2]

shx = 0.3

M = np.float32([[1, shx, 0], [0, 1, 0]])

new_width = int(width + height * shx)
sheared_img = cv2.warpAffine(image, M, (new_width, height))

fig, ax = plt.subplots(1, 2, figsize=(8, 4))
ax[0].imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
ax[0].set_title('Original')
ax[0].axis('off')

ax[1].imshow(cv2.cvtColor(sheared_img, cv2.COLOR_BGR2RGB))
ax[1].set_title('Sheared Image')
ax[1].axis('off')

plt.show()
```

geometreic operations- Affine Transformation

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

image = cv2.imread(r"C:\Users\my.ac.p2mca24004.AMRITAMYSORE\Downloads\tumor.jpg")
image = cv2.resize(image, (300, 300))

height, width = image.shape[:2]

pts1 = np.float32([[0, 0], [width - 1, 0], [0, height - 1]])

pts2 = np.float32([[50, 50], [width - 100, 100], [100, height - 50]])

M = cv2.getAffineTransform(pts1, pts2)

affine_img = cv2.warpAffine(image, M, (width, height))

fig, ax = plt.subplots(1, 2, figsize=(8, 4))
ax[0].imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
ax[0].set_title('Original')
ax[0].axis('off')

ax[1].imshow(cv2.cvtColor(affine_img, cv2.COLOR_BGR2RGB))
ax[1].set_title('Affine Transformation')
ax[1].axis('off')

plt.show()
```

basic Gray Level Transformed

linear transformation type 1: linear identity transformation type 2: linear negative transformation

```python
import cv2
import matplotlib.pyplot as plt
import numpy as np

img = cv2.imread(r"C:\Users\my.ac.p2mca24004.AMRITAMYSORE\Downloads\tumor.jpg")
img = cv2.resize(img, (100, 100))

img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Identity Transformation
identity_transformed = [[0 for _ in range(gray_img.shape[1])] for _ in range(gray_img.shape[0])]
for i in range(gray_img.shape[0]):
    for j in range(gray_img.shape[1]):
        identity_transformed[i][j] = gray_img[i, j]
identity_transformed = np.array(identity_transformed, dtype=np.uint8)

# Negative Transformation
negative_transformed = [[0 for _ in range(gray_img.shape[1])] for _ in range(gray_img.shape[0])]
for i in range(gray_img.shape[0]):
    for j in range(gray_img.shape[1]):
        r = gray_img[i, j]
        s = 255 - r
        negative_transformed[i][j] = s
negative_transformed = np.array(negative_transformed, dtype=np.uint8)

fig, ax = plt.subplots(1, 3, figsize=(9, 3))
ax[0].imshow(img_rgb)
ax[0].set_title('Original Image')
ax[0].axis('off')
ax[1].imshow(identity_transformed, cmap='gray')
ax[1].set_title('Identity Transformed')
ax[1].axis('off')
ax[2].imshow(negative_transformed, cmap='gray')
ax[2].set_title('Negative Transformed')
ax[2].axis('off')
plt.tight_layout()
plt.show()
```

logarithmic - LOG and Inverse Log transformations

```python
import cv2
import matplotlib.pyplot as plt
import math

img = cv2.imread(r"C:\Users\my.ac.p2mca24004.AMRITAMYSORE\Downloads\tumor.jpg")
img = cv2.resize(img, (100, 100))
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

#log transformation
max_val = 0
for i in range(gray_img.shape[0]):
    for j in range(gray_img.shape[1]):
        if gray_img[i, j] > max_val:
            max_val = gray_img[i, j]

c = 255 / math.log(1 + max_val)
log_transformed = gray_img.copy()

for i in range(gray_img.shape[0]):
    for j in range(gray_img.shape[1]):
        r = gray_img[i, j]
        s = c * math.log(1 + r)
        log_transformed[i, j] = int(s)

# inverse log transformation
inverse_log = log_transformed.copy()

for i in range(log_transformed.shape[0]):
    for j in range(log_transformed.shape[1]):
        r = log_transformed[i, j]
        s = math.exp(r / c) - 1
        s = max(0, min(255, s))
        inverse_log[i, j] = int(s)

fig, ax = plt.subplots(1, 3, figsize=(9, 3))
ax[0].imshow(gray_img, cmap='gray')
ax[0].set_title('Original Grayscale')
ax[0].axis('off')
ax[1].imshow(log_transformed, cmap='gray')
ax[1].set_title('Log Transformed')
ax[1].axis('off')
ax[2].imshow(inverse_log, cmap='gray')
ax[2].set_title('Inverse Log')
ax[2].axis('off')
plt.tight_layout()
plt.show()
```

power law transformation -nth power and nth root

```python
import cv2
import matplotlib.pyplot as plt
import numpy as np

img = cv2.imread(r"C:\Users\my.ac.p2mca24004.AMRITAMYSORE\Downloads\tumor.jpg")
img = cv2.resize(img, (100, 100))
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

gamma_power = 2.0
gamma_root = 0.5

nth_power_img = np.zeros_like(gray_img, dtype=np.uint8)
nth_root_img = np.zeros_like(gray_img, dtype=np.uint8)

for i in range(gray_img.shape[0]):
    for j in range(gray_img.shape[1]):
        r = gray_img[i, j] / 255.0

        s_power = r ** gamma_power
        nth_power_img[i, j] = int(s_power * 255)

        s_root = r ** gamma_root
        nth_root_img[i, j] = int(s_root * 255)

fig, ax = plt.subplots(1, 3, figsize=(9, 3))

ax[0].imshow(gray_img, cmap='gray')
ax[0].set_title('Original Grayscale')
ax[0].axis('off')

ax[1].imshow(nth_power_img, cmap='gray')
ax[1].set_title(f'Nth Power (gamma={gamma_power})')
ax[1].axis('off')

ax[2].imshow(nth_root_img, cmap='gray')
ax[2].set_title(f'Nth Root (gamma={gamma_root})')
ax[2].axis('off')

plt.tight_layout()
plt.show()
```

pice-wise types(binary threshold,contrast sretching, gray level sliceing)

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread(r"C:\Users\my.ac.p2mca24004.AMRITAMYSORE\Downloads\tumor.jpg")
img = cv2.resize(img, (100, 100))
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

threshold = 100
binary_thresh = np.zeros_like(gray_img)
for i in range(gray_img.shape[0]):
    for j in range(gray_img.shape[1]):
        if gray_img[i, j] > threshold:
            binary_thresh[i, j] = 255
        else:
            binary_thresh[i, j] = 2

r1, r2 = 70, 140
s1, s2 = 2, 255
contrast_stretched = np.zeros_like(gray_img, dtype=np.uint8)
for i in range(gray_img.shape[0]):
    for j in range(gray_img.shape[1]):
        r = gray_img[i, j]
        if r < r1:
            contrast_stretched[i, j] = s1
        elif r > r2:
            contrast_stretched[i, j] = s2
        else:
            contrast_stretched[i, j] = int(((r - r1) / (r2 - r1)) * (s2 - s1) + s1)

A, B = 100, 150
sliced = np.zeros_like(gray_img)
for i in range(gray_img.shape[0]):
    for j in range(gray_img.shape[1]):
        r = gray_img[i, j]
        if A <= r <= B:
            sliced[i, j] = 255
        else:
            sliced[i, j] = 2

fig, ax = plt.subplots(1, 4, figsize=(12, 3))
ax[0].imshow(gray_img, cmap='gray')
ax[0].set_title('Original Grayscale')
ax[0].axis('on')
ax[1].imshow(binary_thresh, cmap='gray', vmin=0, vmax=255)
ax[1].set_title('Binary Thresholding')
ax[1].axis('on')
ax[2].imshow(contrast_stretched, cmap='gray', vmin=0, vmax=255)
ax[2].set_title('Contrast Stretching')
ax[2].axis('on')
ax[3].imshow(sliced, cmap='gray', vmin=0, vmax=255)
ax[3].set_title('Gray Level Slicing')
ax[3].axis('on')
plt.tight_layout()
plt.show()
```

Smoothening filter

mean fliter- mean Coverage filter

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread(r"C:\Users\my.ac.p2mca24004.AMRITAMYSORE\Downloads\tumor.jpg")
img = cv2.resize(img, (100, 100))
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

height, width = gray_img.shape
mean_filtered_img = np.zeros_like(gray_img)

for i in range(1, height - 1):
    for j in range(1, width - 1):
        neighborhood = gray_img[i-1:i+2, j-1:j+2]
        average = np.sum(neighborhood) // 9
        mean_filtered_img[i, j] = average

fig, ax = plt.subplots(1, 2, figsize=(8, 4))
ax[0].imshow(gray_img, cmap='gray')
ax[0].set_title('Original Grayscale')
ax[0].axis('off')

ax[1].imshow(mean_filtered_img, cmap='gray')
ax[1].set_title('Mean Filtered')
ax[1].axis('off')

plt.tight_layout()
plt.show()
```

Smoothening filters

mean fliter- Weighted average filtre

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread(r"C:\Users\my.ac.p2mca24004.AMRITAMYSORE\Downloads\tumor.jpg")
img = cv2.resize(img, (100, 100))
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

height, width = gray_img.shape
weighted_filtered_img = np.zeros_like(gray_img)

kernel = np.array([[1, 2, 1],
                   [2, 4, 2],
                   [1, 2, 1]])
kernel_sum = np.sum(kernel)

for i in range(1, height - 1):
    for j in range(1, width - 1):
        neighborhood = gray_img[i-1:i+2, j-1:j+2]
        weighted_sum = np.sum(neighborhood * kernel)
        new_pixel_value = weighted_sum // kernel_sum
        weighted_filtered_img[i, j] = new_pixel_value

fig, ax = plt.subplots(1, 2, figsize=(8, 4))
ax[0].imshow(gray_img, cmap='gray')
ax[0].set_title('Original Grayscale')
ax[0].axis('off')

ax[1].imshow(weighted_filtered_img, cmap='gray')
ax[1].set_title('Weighted Average Filtered')
ax[1].axis('off')

plt.tight_layout()
plt.show()
```

Smoothening filters - order statistical filter (min,max,median)

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread(r"C:\Users\my.ac.p2mca24004.AMRITAMYSORE\Downloads\tumor.jpg")
img = cv2.resize(img, (100, 100))
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

height, width = img.shape
min_filtered_img = np.zeros_like(img)
max_filtered_img = np.zeros_like(img)
median_filtered_img = np.zeros_like(img)

kernel_size = 3
pad = kernel_size // 2

padded_img = cv2.copyMakeBorder(img, pad, pad, pad, pad, cv2.BORDER_REPLICATE)

for i in range(height):
    for j in range(width):
        neighborhood = []
        for k in range(kernel_size):
            for l in range(kernel_size):
                neighborhood.append(padded_img[i + k, j + l])

        neighborhood.sort()

        min_filtered_img[i, j] = neighborhood[0]
        max_filtered_img[i, j] = neighborhood[-1]
        median_filtered_img[i, j] = neighborhood[len(neighborhood) // 2]

fig, ax = plt.subplots(1, 4, figsize=(16, 4))

ax[0].imshow(img, cmap='gray', vmin=0, vmax=255)
ax[0].set_title('Original Grayscale Image')
ax[0].axis('off')

ax[1].imshow(min_filtered_img, cmap='gray', vmin=0, vmax=255)
ax[1].set_title('Min Filter')
ax[1].axis('off')

ax[2].imshow(max_filtered_img, cmap='gray', vmin=0, vmax=255)
ax[2].set_title('Max Filter')
ax[2].axis('off')

ax[3].imshow(median_filtered_img, cmap='gray', vmin=0, vmax=255)
ax[3].set_title('Median Filter')
ax[3].axis('off')

plt.tight_layout()
plt.show()
```

sharpening filter

first order - First order: Robert

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread(r"C:\Users\my.ac.p2mca24004.AMRITAMYSORE\Downloads\tumor.jpg")
img = cv2.resize(img, (100, 100))
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

kernel_x = np.array([[1, 0], [0, -1]], dtype=np.float32)
kernel_y = np.array([[0, 1], [-1, 0]], dtype=np.float32)

height, width = gray_img.shape
robert_x = np.zeros_like(gray_img, dtype=np.float32)
robert_y = np.zeros_like(gray_img, dtype=np.float32)

padded_img = np.pad(gray_img, 1, mode='edge')

for i in range(height):
    for j in range(width):
        robert_x[i, j] = np.sum(kernel_x * padded_img[i:i+2, j:j+2])
        robert_y[i, j] = np.sum(kernel_y * padded_img[i:i+2, j:j+2])

edge_magnitude = np.abs(robert_x) + np.abs(robert_y)
edge_magnitude = np.clip(edge_magnitude, 0, 255).astype(np.uint8)

sharpened_img = cv2.add(gray_img, edge_magnitude)

fig, ax = plt.subplots(1, 3, figsize=(12, 4))
ax[0].imshow(gray_img, cmap='gray')
ax[0].set_title('Original Image')
ax[0].axis('off')
ax[1].imshow(edge_magnitude, cmap='gray')
ax[1].set_title('Robert Edges')
ax[1].axis('off')
ax[2].imshow(sharpened_img, cmap='gray')
ax[2].set_title('Robert Sharpened')
ax[2].axis('off')
plt.tight_layout()
plt.show()
```

sharpening filter

first order - First order: Sobel

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread(r"C:\Users\my.ac.p2mca24004.AMRITAMYSORE\Downloads\tumor.jpg")
img = cv2.resize(img, (100, 100))
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

kernel_x = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]], dtype=np.float32)
kernel_y = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]], dtype=np.float32)

height, width = gray_img.shape
sobel_x = np.zeros_like(gray_img, dtype=np.float32)
sobel_y = np.zeros_like(gray_img, dtype=np.float32)

padded_img = np.pad(gray_img, 1, mode='edge')

for i in range(height):
    for j in range(width):
        sobel_x[i, j] = np.sum(kernel_x * padded_img[i:i+3, j:j+3])
        sobel_y[i, j] = np.sum(kernel_y * padded_img[i:i+3, j:j+3])

edge_magnitude = np.abs(sobel_x) + np.abs(sobel_y)
edge_magnitude = np.clip(edge_magnitude, 0, 255).astype(np.uint8)

sharpened_img = cv2.add(gray_img, edge_magnitude)

fig, ax = plt.subplots(1, 3, figsize=(12, 4))
ax[0].imshow(gray_img, cmap='gray')
ax[0].set_title('Original Image')
ax[0].axis('off')
ax[1].imshow(edge_magnitude, cmap='gray')
ax[1].set_title('Sobel Edges')
ax[1].axis('off')
ax[2].imshow(sharpened_img, cmap='gray')
ax[2].set_title('Sobel Sharpened')
ax[2].axis('off')
plt.tight_layout()
plt.show()
```

sharpening filter

first order - First order: Laplacian

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread(r"C:\Users\my.ac.p2mca24004.AMRITAMYSORE\Downloads\tumor.jpg")
img = cv2.resize(img, (100, 100))
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

kernel = np.array([[0, 1, 0], [1, -4, 1], [0, 1, 0]], dtype=np.float32)

height, width = gray_img.shape
laplacian_output = np.zeros_like(gray_img, dtype=np.float32)

padded_img = np.pad(gray_img, 1, mode='edge')

for i in range(height):
    for j in range(width):
        laplacian_output[i, j] = np.sum(kernel * padded_img[i:i+3, j:j+3])

laplacian_image = np.clip(np.abs(laplacian_output), 0, 255).astype(np.uint8)

sharpened_img_float = gray_img.astype(np.float32) - laplacian_output
sharpened_img = np.clip(sharpened_img_float, 0, 255).astype(np.uint8)

fig, ax = plt.subplots(1, 3, figsize=(12, 4))
ax[0].imshow(gray_img, cmap='gray')
ax[0].set_title('Original Image')
ax[0].axis('off')
ax[1].imshow(laplacian_image, cmap='gray')
ax[1].set_title('Laplacian Edges')
ax[1].axis('off')
ax[2].imshow(sharpened_img, cmap='gray')
ax[2].set_title('Laplacian Sharpened')
ax[2].axis('off')
plt.tight_layout()
plt.show()
```

sharpening filter

second order - Laplacian

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread(r"C:\Users\my.ac.p2mca24004.AMRITAMYSORE\Downloads\tumor.jpg")
img = cv2.resize(img, (100, 100))
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

kernel = np.array([[1, 1, 1], [1, -8, 1], [1, 1, 1]], dtype=np.float32)

height, width = gray_img.shape
laplacian_output = np.zeros_like(gray_img, dtype=np.float32)

padded_img = np.pad(gray_img, 1, mode='edge')

for i in range(height):
    for j in range(width):
        laplacian_output[i, j] = np.sum(kernel * padded_img[i:i+3, j:j+3])

laplacian_image = np.clip(np.abs(laplacian_output), 0, 255).astype(np.uint8)

sharpened_img_float = gray_img.astype(np.float32) - laplacian_output
sharpened_img = np.clip(sharpened_img_float, 0, 255).astype(np.uint8)

fig, ax = plt.subplots(1, 3, figsize=(12, 4))
ax[0].imshow(gray_img, cmap='gray')
ax[0].set_title('Original Image')
ax[0].axis('off')

ax[1].imshow(laplacian_image, cmap='gray')
ax[1].set_title('Laplacian Edges (8-conn)')
ax[1].axis('off')

ax[2].imshow(sharpened_img, cmap='gray')
ax[2].set_title('Laplacian Sharpened (8-conn)')
ax[2].axis('off')

plt.tight_layout()
plt.show()
```