**نمونه کدهای خروجی:**

نمونه کدهای زیر برای حالت Blurring با ورودی های Averaging Blurring و Median Blurring وGaussian Blurring آورده شده است.

و در ادامه نمونه کد برای حالت بهبود Contrast آورده شده است.

در انتها نیز تعدادی مثال های ورودی و خروجی نتابج آورده شده است.

Averaging Blurring

def Averaging\_Blurring(img, filter\_size):

    # Reflect101 padding

    image = Reflect101(img, filter\_size)

    # Initialize result array

    result = np.zeros\_like(img)

    # Compute averaging blurring

    filter\_height =filter\_size

    filter\_width = filter\_size

    for i in range(result.shape[0]):

      for j in range(result.shape[1]):

        result[i, j] = np.mean(image[i:i+filter\_height, j:j+filter\_width])

    return result

Median Blurring

def Median\_Blurring(img, filter\_size):

    # Reflect101 padding

    image = Reflect101(img, filter\_size)

    # Compute filter dimensions

    filter\_height = filter\_size

    filter\_width = filter\_size

    # Calculate the median pixel value in each filter region

    result = np.median(np.array([

        [

          image[i:i+filter\_height, j:j+filter\_width]

          for j in range(image.shape[1] - filter\_width + 1)

        ]

        for i in range(image.shape[0] - filter\_height + 1)

    ]), axis=(2, 3))

    # By specifying axis=(2, 3), we're essentially telling NumPy to calculate the median across

    # the height and width dimensions of each filter region.

    return result

Gaussian Blurring

def Gaussian\_Blurring(img, filter\_size, std):

    # Compute the kernel size

    kernel\_radius = filter\_size // 2

    # Generate Gaussian kernel

    x = np.arange(-kernel\_radius, kernel\_radius + 1)

    y = np.arange(-kernel\_radius, kernel\_radius + 1)

    xx, yy = np.meshgrid(x, y)

    kernel = np.exp(-(xx\*\*2 + yy\*\*2) / (2 \* std\*\*2)) / (2 \* np.pi \* std\*\*2)

    kernel /= np.sum(kernel)  # Normalize the kernel

    # Reflect101 padding

    output = Reflect101(img, filter\_size)

    # Apply Gaussian blur using convolution

    result = np.zeros\_like(img)

    for i in range(result.shape[0]):

        for j in range(result.shape[1]):

            result[i, j] = np.sum(output[i:i+filter\_size, j:j+filter\_size] \* kernel)

    return result

Contrast

def stretch\_hist(image):

  output\_image = image.copy()

  # Start

  Min\_Pixel\_Value = np.min(image)

  Max\_Pixel\_Value = np.max(image)

  New\_Min = 0

  New\_Max = 255

  for row in output\_image:

    for i in range(len(row)):

      x = ((row[i]-Min\_Pixel\_Value)/(Max\_Pixel\_Value-Min\_Pixel\_Value)) \* (New\_Max-New\_Min) + New\_Min

      row[i] = x

  # End

  return output\_image



