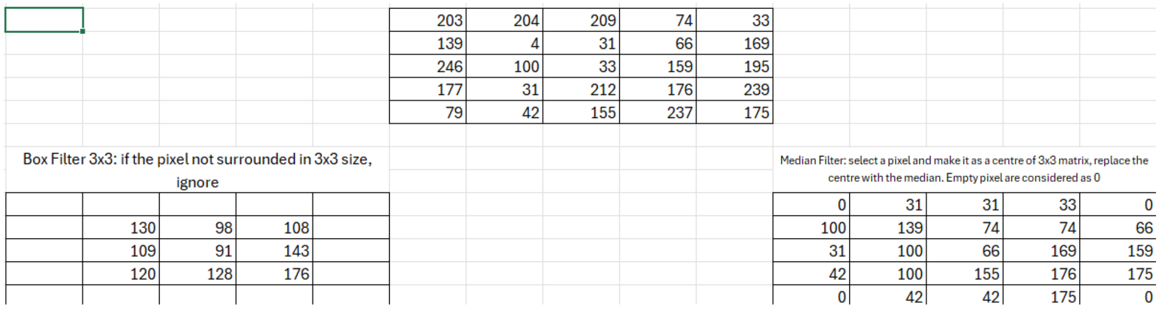
Alethea Agung Yodha Pratama - 2602078732

1. What is the concept of image processing, its importance in computer vision, and the role of preprocessing techniques like resizing, normalization, and augmentation in enhancing image datasets to contribute to better performance in machine learning models?

Image processing involves techniques to improve or analyze images for better understanding, crucial in computer vision as it helps interpret visual data effectively. Preprocessing techniques like resizing, normalization, and augmentation prepare images for analysis: resizing adjusts dimensions for consistency, normalization scales pixel values, and augmentation creates variations to increase dataset size. These processes refine data quality, leading to more accurate and efficient machine learning models by improving feature extraction and model training.

2. Compare and contrast two common image filtering techniques: Box filter and Median filter, and apply them using a 5x5 kernel.



203	204	209	74	33
139	4	31	66	169
246	100	33	159	195
177	31	212	176	239
79	42	155	237	175

Box Filter 3x3: if the pixel not surrounded in 3x3 size, ignore

	130	98	108	
	109	91	143	
	120	128	176	

Median Filter: select a pixel and make it as a centre of 3x3 matrix, replace the centre with the median. Empty pixel are considered as 0

0	31	31	33	0
100	139	74	74	66
31	100	66	169	159
42	100	155	176	175
0	42	42	175	0

All calculations are done in excel, borders are ignored when counting the box filter

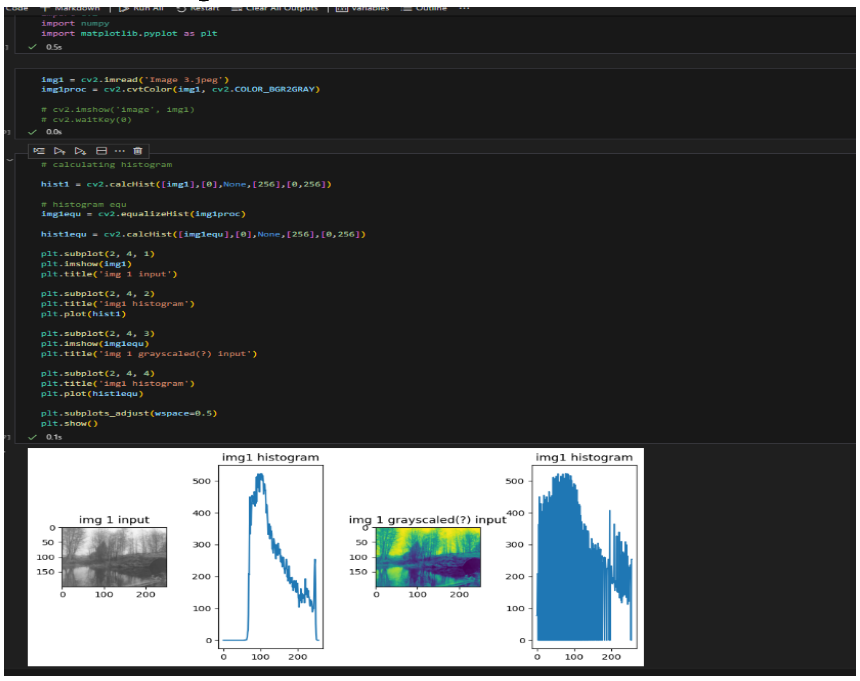
Box Filter: Smooths the image by averaging the pixel values within the kernel, reducing noise but also potentially blurring details. Median Filter: Replaces each pixel with the median value within the kernel, effectively removing noise while preserving edges better than the box filter. The box filter yields an average value of approximately 104.04, while the median filter yields a median value of 113.

3. Feature detection is essential in computer vision for identifying key points of interest in images. Choose a feature detection algorithm (e.g., SIFT, Harris Corner Detector, or ORB) and describe: The steps involved in detecting features using the chosen algorithm, The types of features the algorithm is best suited for, Advantages and limitations of the chosen method in real-world applications like object tracking or recognition.

SIFT (Scale-Invariant Feature Transform):
- Steps: Scale-space extrema detection, Keypoint localization, Orientation assignment, Keypoint descriptor, Matching.
- Best suited for: Scale and rotation invariant features.
- Advantages: Robust to scaling, rotation, and noise; effective for object recognition.
- Limitations: Computationally intensive; patent restrictions.

Code

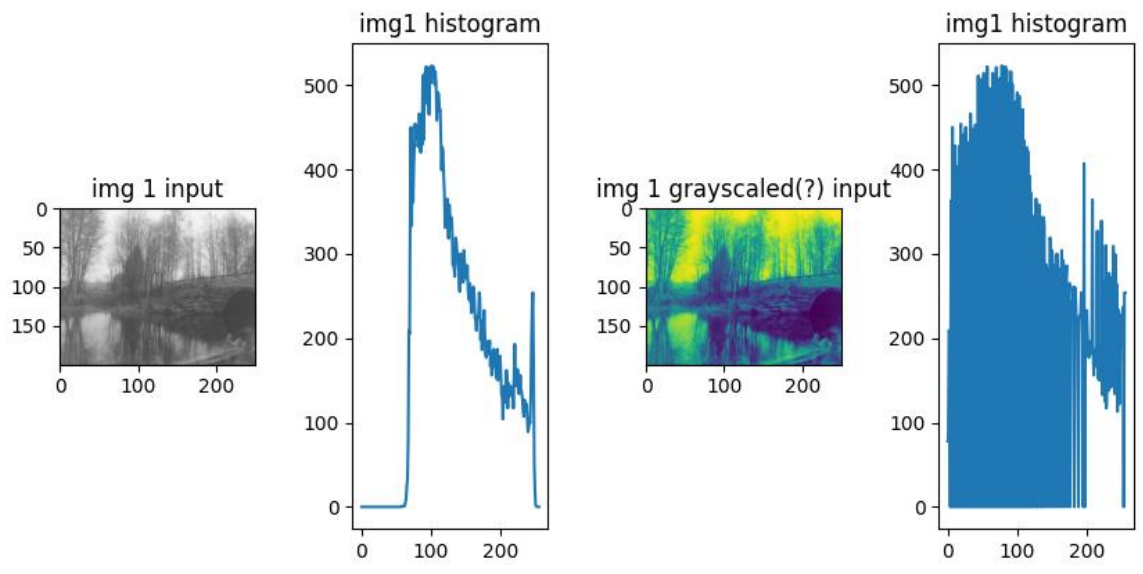
1.
   a. Code and histogram



```python
import numpy
import matplotlib.pyplot as plt
```
✓ 0.5s

```python
img1 = cv2.imread('Image 3.jpeg')
img1proc = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)

# cv2.imshow('image', img1)
# cv2.waitKey(0)
```
✓ 0.0s

```python
# calculating histogram

hist1 = cv2.calcHist([img1],[0],None,[256],[0,256])

# histogram equ
img1equ = cv2.equalizeHist(img1proc)

hist1equ = cv2.calcHist([img1equ],[0],None,[256],[0,256])

plt.subplot(2, 4, 1)
plt.imshow(img1)
plt.title('img 1 input')

plt.subplot(2, 4, 2)
plt.title('img1 histogram')
plt.plot(hist1)

plt.subplot(2, 4, 3)
plt.imshow(img1equ)
plt.title('img 1 grayscaled(?) input')

plt.subplot(2, 4, 4)
plt.title('img1 histogram')
plt.plot(hist1equ)

plt.subplots_adjust(wspace=0.5)
plt.show()
```
✓ 0.1s



b. 1. Contrast Improvement:

   - Effect: Histogram equalization redistributes the pixel intensity values across the entire range of possible values (0 to 255 for an 8-bit image). This process enhances the contrast, making the details in both the dark and bright areas more visible.

   - Result: After histogram equalization, the image appears more dynamic, with enhanced detail and contrast that were previously not as distinguishable.

   2. Noticeable Artifacts or Limitations:

   - Artifacts: Sometimes, histogram equalization can introduce artifacts such as a noisy or grainy appearance, especially in regions of the image with uniform intensity. This happens because the algorithm might amplify small variations in pixel intensities, making them more noticeable.

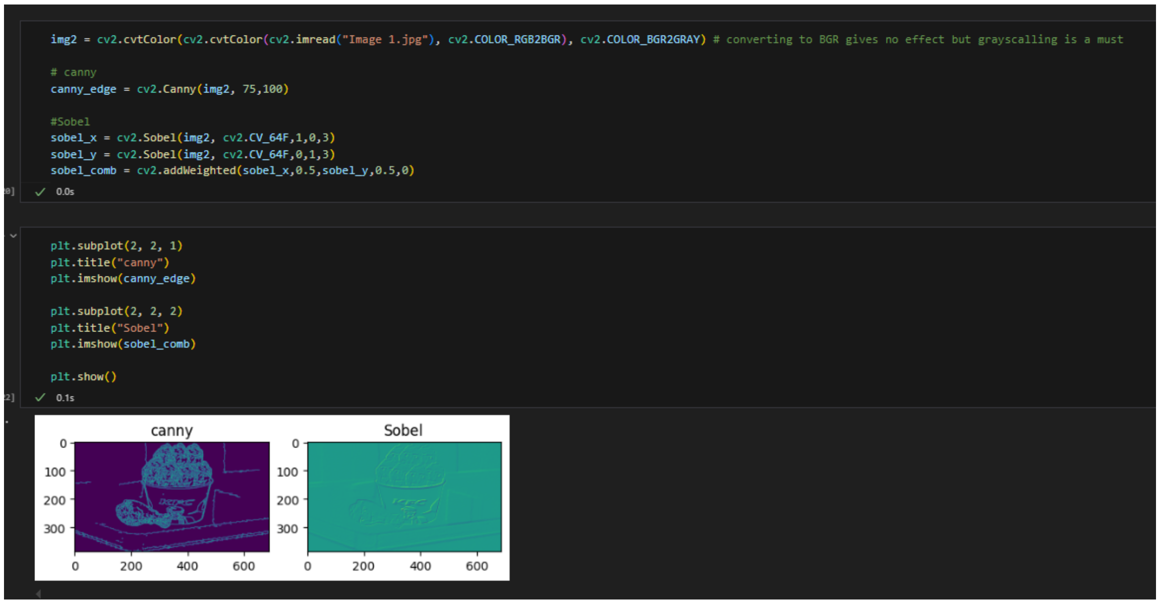
   - Limitations:

     + Over-Enhancement: In some cases, histogram equalization can lead to over-enhancement, where the contrast is increased too much, making the image look unnatural.

     + Loss of Detail: If the original image has large areas of similar intensity, histogram equalization might not effectively enhance the contrast for those areas, leading to potential loss of detail.

2.

a. Code, result, and input

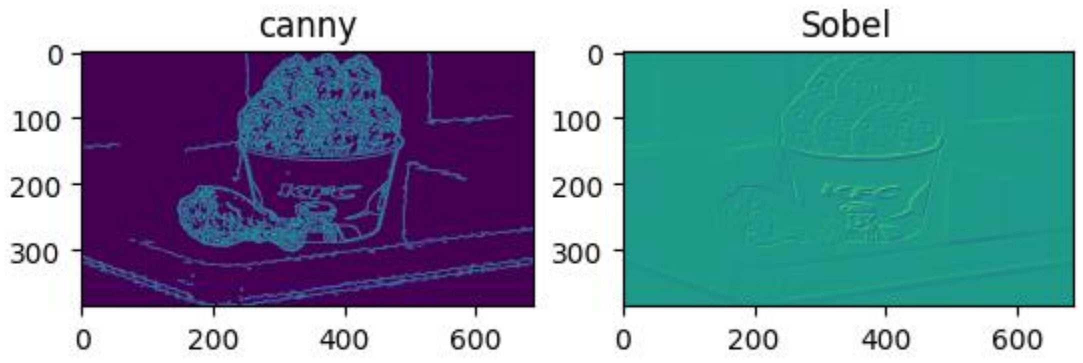


```
img2 = cv2.cvtColor(cv2.cvtColor(cv2.imread("Image 1.jpg"), cv2.COLOR_RGB2BGR), cv2.COLOR_BGR2GRAY) # converting to BGR gives no effect but grayscalling is a must

# canny
canny_edge = cv2.Canny(img2, 75,100)

#Sobel
sobel_x = cv2.Sobel(img2, cv2.CV_64F,1,0,3)
sobel_y = cv2.Sobel(img2, cv2.CV_64F,0,1,3)
sobel_comb = cv2.addWeighted(sobel_x,0.5,sobel_y,0.5,0)
```

✓ 0.0s

```
plt.subplot(2, 2, 1)
plt.title("canny")
plt.imshow(canny_edge)

plt.subplot(2, 2, 2)
plt.title("Sobel")
plt.imshow(sobel_comb)

plt.show()
```

✓ 0.1s

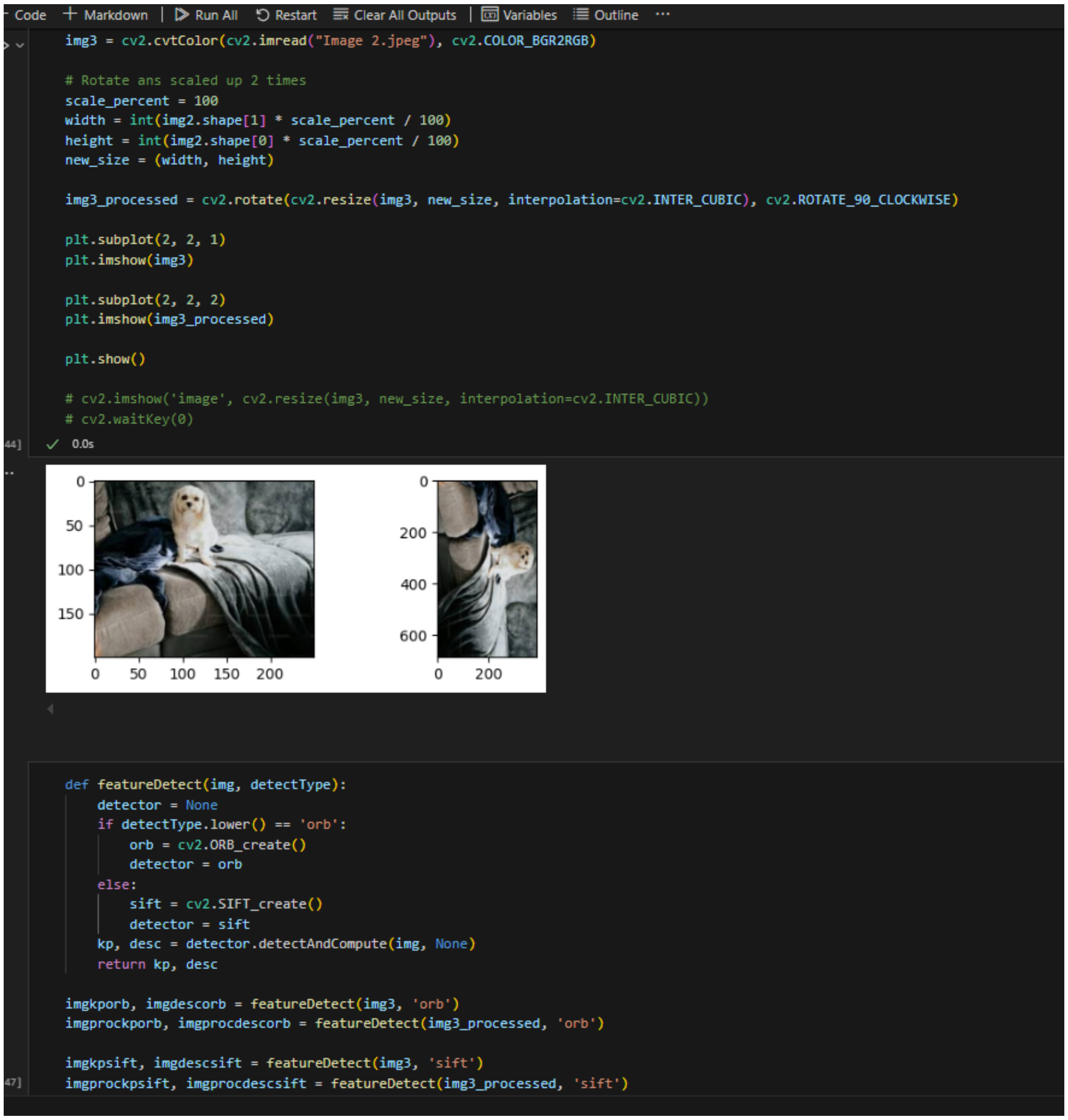


b. The Canny edge detection method provides clearer edges for this image. This is because the Canny method uses a multi-stage process, including noise reduction, gradient calculation, non-maximum suppression, and edge tracking by hysteresis. These steps help to accurately and distinctly detect a wide range of edges, making them more defined and clear.

   On the other hand, the Sobel method, while effective for detecting edges, tends to produce less sharp and more gradient-like edges. It calculates the gradient of image intensity at each pixel, highlighting regions of high spatial frequency corresponding to edges, but lacks the additional refinement steps that the Canny method includes.

3.

Code, result, and input

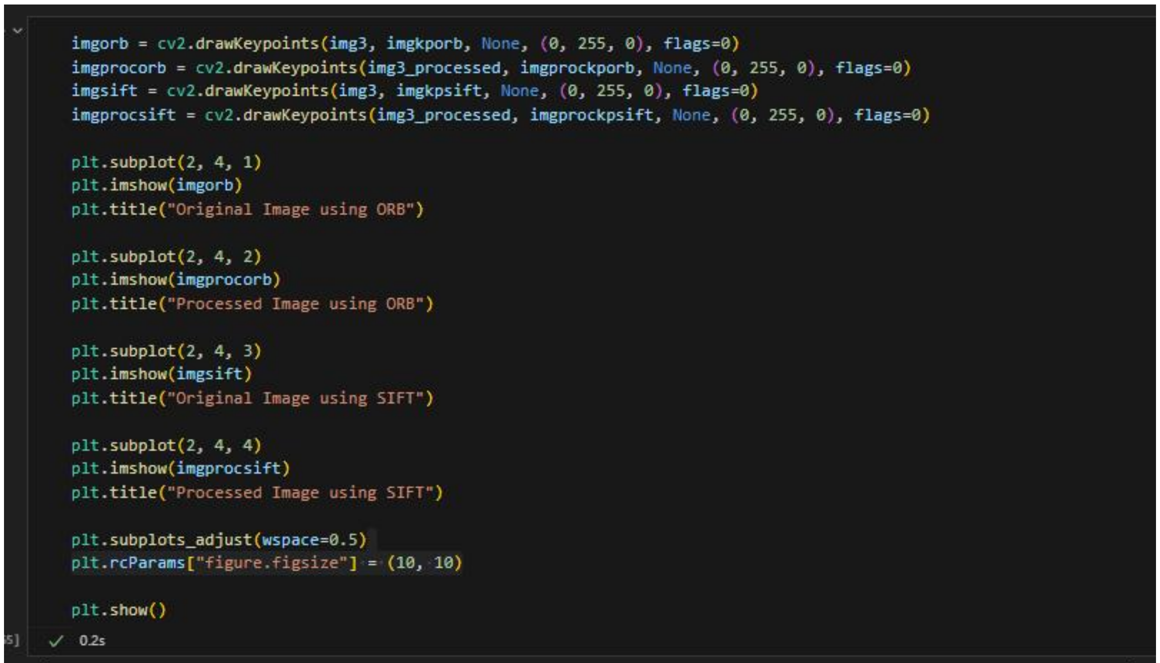


```python
img3 = cv2.cvtColor(cv2.imread("Image 2.jpeg"), cv2.COLOR_BGR2RGB)

# Rotate ans scaled up 2 times
scale_percent = 100
width = int(img2.shape[1] * scale_percent / 100)
height = int(img2.shape[0] * scale_percent / 100)
new_size = (width, height)

img3_processed = cv2.rotate(cv2.resize(img3, new_size, interpolation=cv2.INTER_CUBIC), cv2.ROTATE_90_CLOCKWISE)

plt.subplot(2, 2, 1)
plt.imshow(img3)

plt.subplot(2, 2, 2)
plt.imshow(img3_processed)

plt.show()

# cv2.imshow('image', cv2.resize(img3, new_size, interpolation=cv2.INTER_CUBIC))
# cv2.waitKey(0)
```

44] ✓ 0.0s

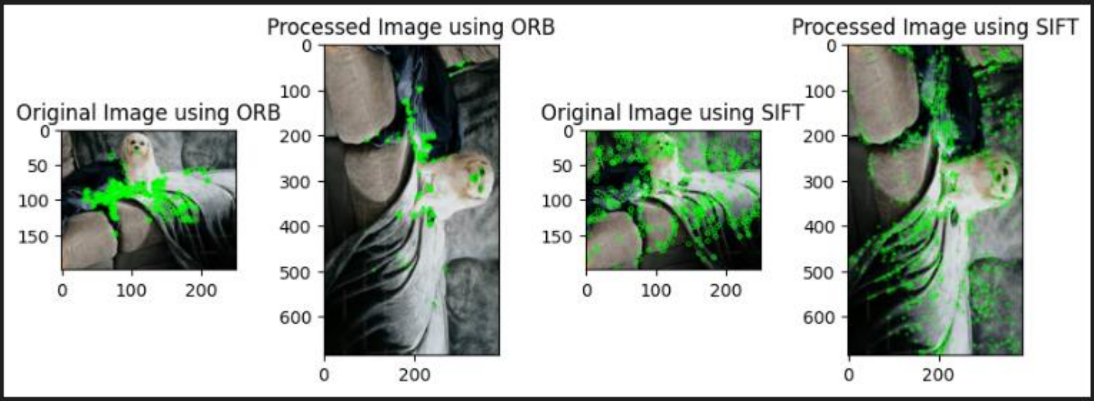
```python
def featureDetect(img, detectType):
    detector = None
    if detectType.lower() == 'orb':
        orb = cv2.ORB_create()
        detector = orb
    else:
        sift = cv2.SIFT_create()
        detector = sift
    kp, desc = detector.detectAndCompute(img, None)
    return kp, desc

imgkporb, imgdescorb = featureDetect(img3, 'orb')
imgprockporb, imgprocdescorb = featureDetect(img3_processed, 'orb')

imgkpsift, imgdescsift = featureDetect(img3, 'sift')
imgprockpsift, imgprocdescsift = featureDetect(img3_processed, 'sift')
```

47]



```
imgorb = cv2.drawKeypoints(img3, imgkporb, None, (0, 255, 0), flags=0)
imgprocorb = cv2.drawKeypoints(img3_processed, imgprockporb, None, (0, 255, 0), flags=0)
imgsift = cv2.drawKeypoints(img3, imgkpsift, None, (0, 255, 0), flags=0)
imgprocsift = cv2.drawKeypoints(img3_processed, imgprockpsift, None, (0, 255, 0), flags=0)

plt.subplot(2, 4, 1)
plt.imshow(imgorb)
plt.title("Original Image using ORB")

plt.subplot(2, 4, 2)
plt.imshow(imgprocorb)
plt.title("Processed Image using ORB")

plt.subplot(2, 4, 3)
plt.imshow(imgsift)
plt.title("Original Image using SIFT")

plt.subplot(2, 4, 4)
plt.imshow(imgprocsift)
plt.title("Processed Image using SIFT")

plt.subplots_adjust(wspace=0.5)
plt.rcParams["figure.figsize"] = (10, 10)

plt.show()
```

```
✓ 0.2s
```


Original Image using ORB
Processed Image using ORB
Original Image using SIFT
Processed Image using SIFT
0
50
100
150
0
100
200
300
400
500
600
200

Full code: https://github.com/yodhasu/utsComVis2024