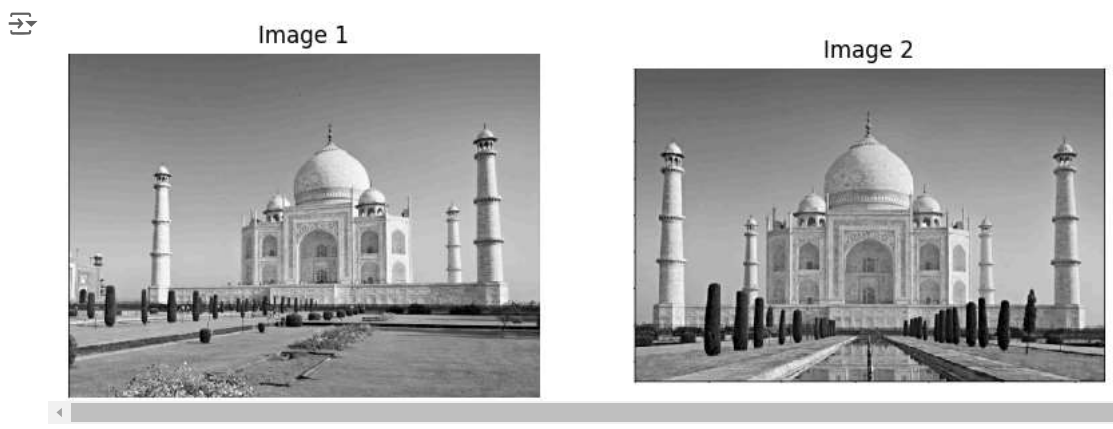
NAME:SAIM CHISHTI

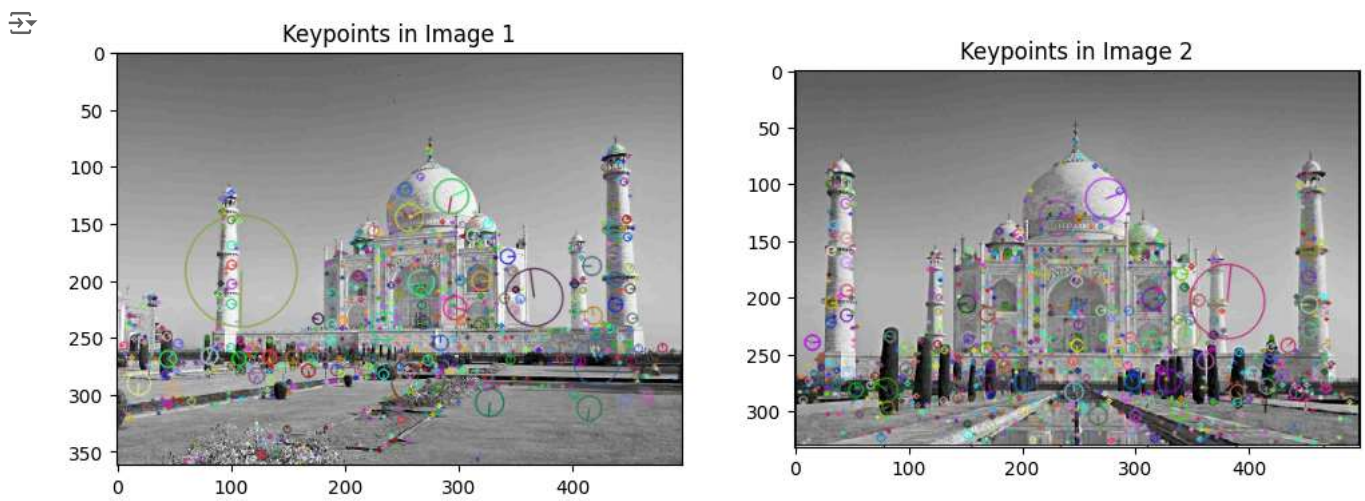
CLASS:BSAI(6A)

ENROLLMENT:01-136221-045

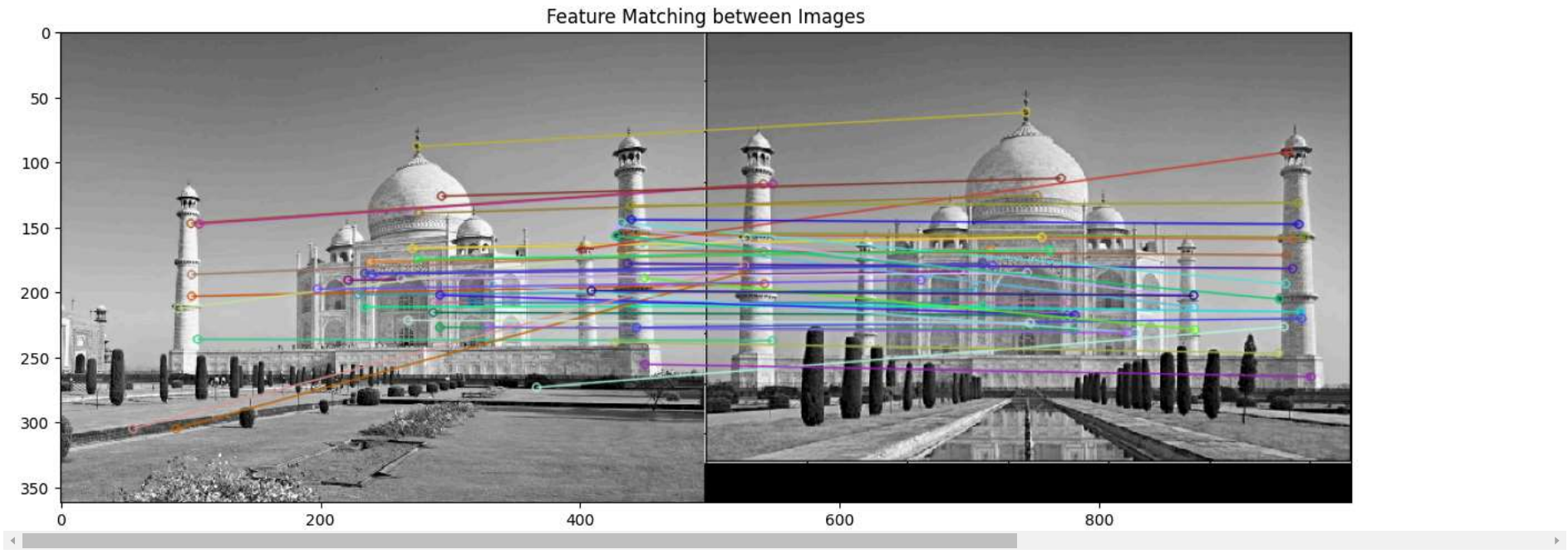
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
# Load two images for feature detection and matching
img1 = cv2.imread('image1.png', cv2.IMREAD_GRAYSCALE)  # First image
img2 = cv2.imread('Image2.png', cv2.IMREAD_GRAYSCALE)  # Second image
plt.figure(figsize=(10, 5))

# Show the first image
plt.subplot(1, 2, 1)
plt.imshow(img1, cmap='gray')
plt.title('Image 1')
plt.axis('off')  # Hide axis

# Show the second image
plt.subplot(1, 2, 2)
plt.imshow(img2, cmap='gray')
plt.title('Image 2')
plt.axis('off')  # Hide axis

# Display the images
plt.show()
```




```
# Create a SIFT detector
sift = cv2.SIFT_create()
# Detect keypoints and compute descriptors for both images
keypoints1, descriptors1 = sift.detectAndCompute(img1, None)
keypoints2, descriptors2 = sift.detectAndCompute(img2, None)



# Draw keypoints on the images
img1_with_keypoints = cv2.drawKeypoints(img1, keypoints1, None, flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
img2_with_keypoints = cv2.drawKeypoints(img2, keypoints2, None, flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

# Display the keypoints
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.imshow(img1_with_keypoints, cmap='gray')
plt.title('Keypoints in Image 1')

plt.subplot(1, 2, 2)
plt.imshow(img2_with_keypoints, cmap='gray')
plt.title('Keypoints in Image 2')

plt.show()
```

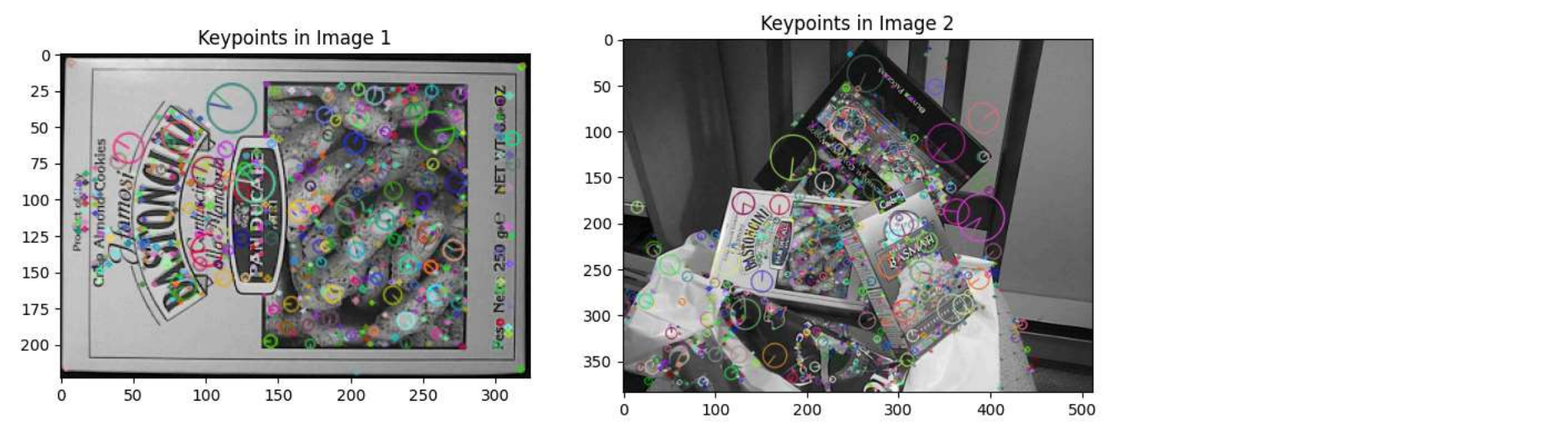



```
# Initialize the BFMatcher
bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)

# Match descriptors
matches = bf.match(descriptors1, descriptors2)

# Sort matches based on their distance (lower distance is better)
matches = sorted(matches, key=lambda x: x.distance)



# Draw the first 50 matches
img_matches = cv2.drawMatches(img1, keypoints1, img2, keypoints2, matches[:50], None, flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)

# Display the matches
plt.figure(figsize=(15, 10))
plt.imshow(img_matches)
plt.title('Feature Matching between Images')
plt.show()
```



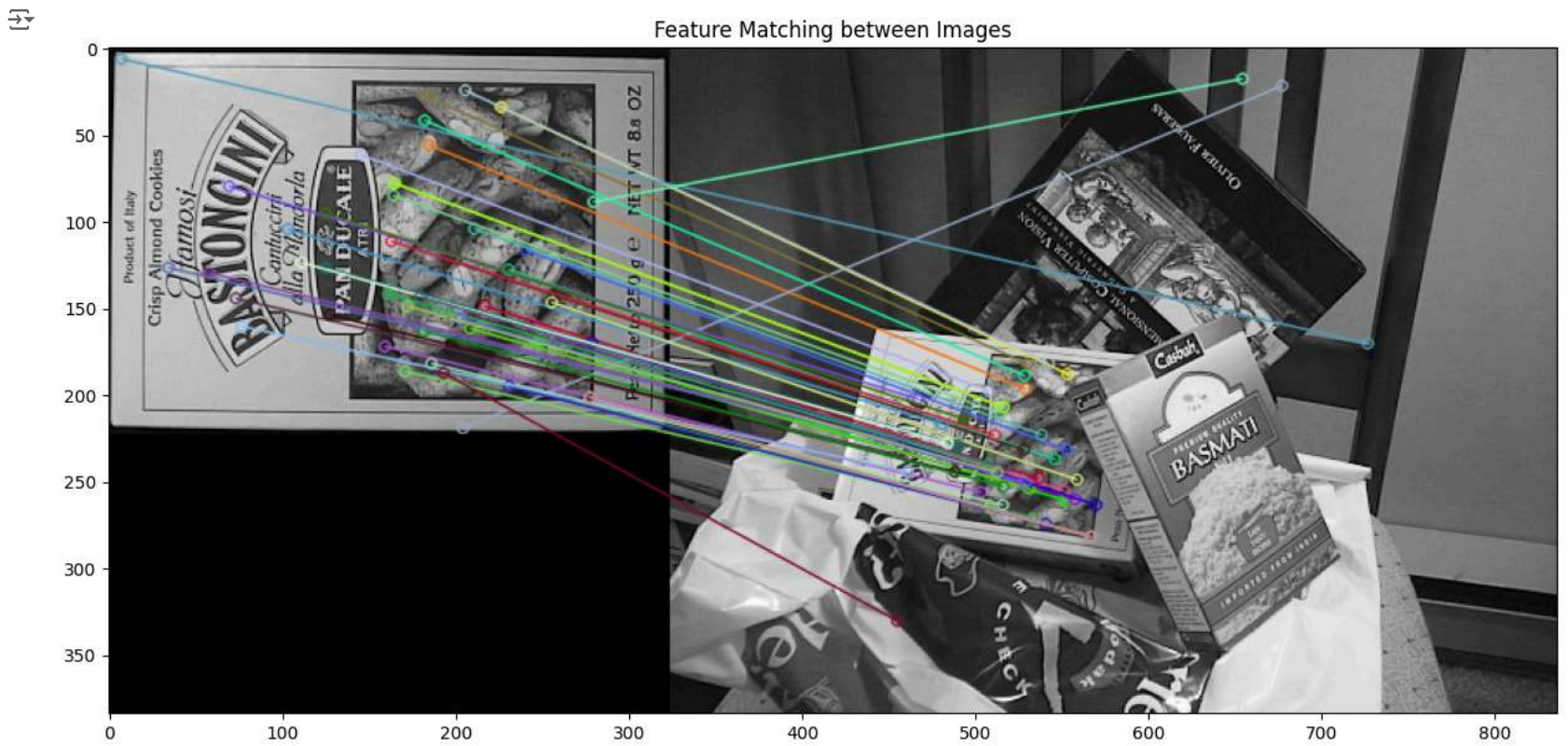

```
img1 = cv2.imread('box_train_image.png', cv2.IMREAD_GRAYSCALE)  # First image
img2 = cv2.imread('box_in_scene_test_image.png', cv2.IMREAD_GRAYSCALE)  # Second image
plt.figure(figsize=(10, 5))

# Show the first image
plt.subplot(1, 2, 1)
plt.imshow(img1, cmap='gray')
plt.title('Image 1')
plt.axis('off')  # Hide axis

# Show the second image
plt.subplot(1, 2, 2)
plt.imshow(img2, cmap='gray')
plt.title('Image 2')
plt.axis('off')  # Hide axis

# Display the images
plt.show()
```


```
# Create a SIFT detector
sift = cv2.SIFT_create()
# Detect keypoints and compute descriptors for both images
keypoints1, descriptors1 = sift.detectAndCompute(img1, None)
keypoints2, descriptors2 = sift.detectAndCompute(img2, None)



# Draw keypoints on the images
img1_with_keypoints = cv2.drawKeypoints(img1, keypoints1, None, flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
img2_with_keypoints = cv2.drawKeypoints(img2, keypoints2, None, flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

# Display the keypoints
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.imshow(img1_with_keypoints, cmap='gray')
plt.title('Keypoints in Image 1')
plt.subplot(1, 2, 2)
plt.imshow(img2_with_keypoints, cmap='gray')
plt.title('Keypoints in Image 2')

plt.show()
```

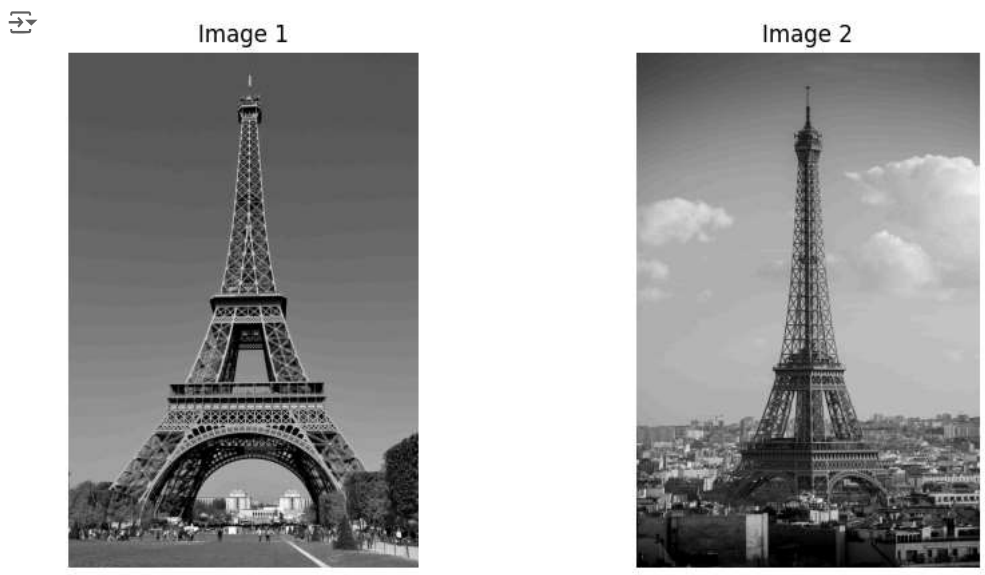



```
# Initialize the BFMatcher
bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)

# Match descriptors
matches = bf.match(descriptors1, descriptors2)

# Sort matches based on their distance (lower distance is better)
matches = sorted(matches, key=lambda x: x.distance)
# Draw the first 50 matches
img_matches = cv2.drawMatches(img1, keypoints1, img2, keypoints2, matches[:50], None, flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
```

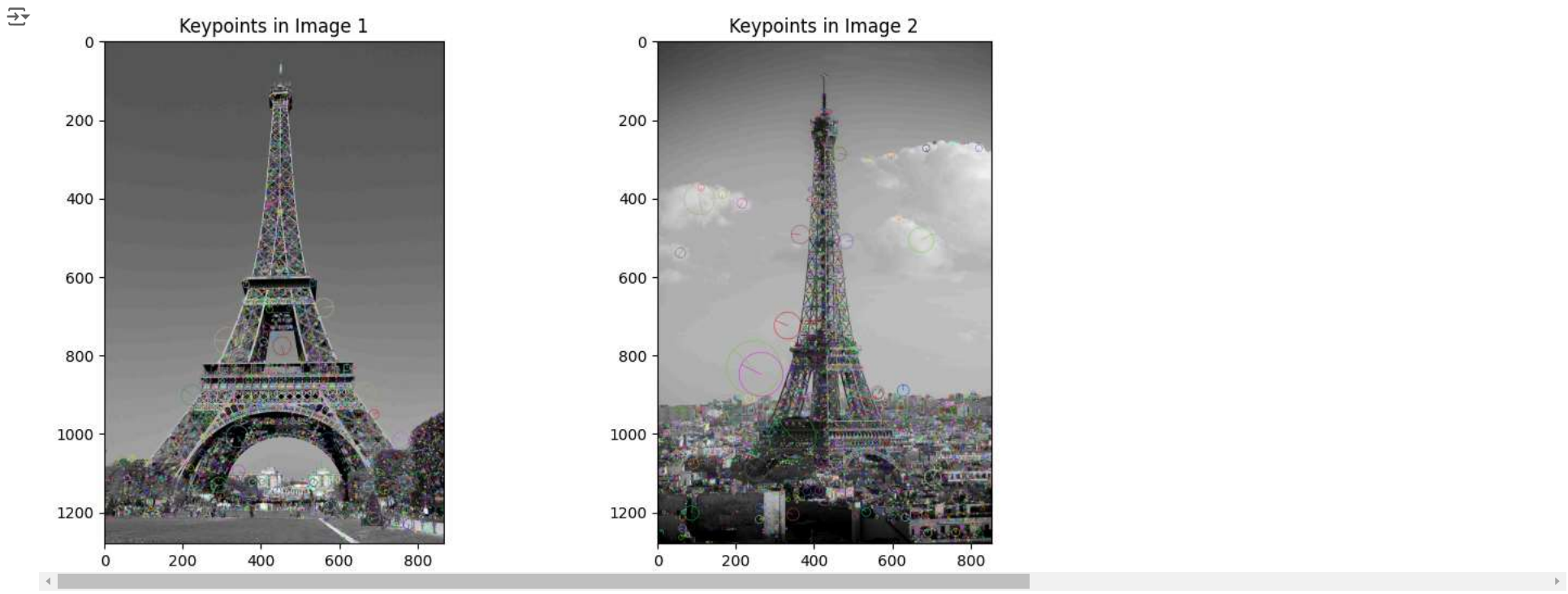
```
# Display the matches
plt.figure(figsize=(15, 10))
plt.imshow(img_matches)
plt.title('Feature Matching between Images')
plt.show()
```




```
img1 = cv2.imread('eiffle_train_image.jpg', cv2.IMREAD_GRAYSCALE)  # First image
img2 = cv2.imread('eiffle_test_image.jpg', cv2.IMREAD_GRAYSCALE)  # Second image
plt.figure(figsize=(10, 5))

# Show the first image
plt.subplot(1, 2, 1)
plt.imshow(img1, cmap='gray')
plt.title('Image 1')
plt.axis('off')  # Hide axis

# Show the second image
plt.subplot(1, 2, 2)
plt.imshow(img2, cmap='gray')
plt.title('Image 2')
plt.axis('off')  # Hide axis

# Display the images
plt.show()
```




```
# Create a SIFT detector
sift = cv2.SIFT_create()
# Detect keypoints and compute descriptors for both images
keypoints1, descriptors1 = sift.detectAndCompute(img1, None)
keypoints2, descriptors2 = sift.detectAndCompute(img2, None)



# Draw keypoints on the images
img1_with_keypoints = cv2.drawKeypoints(img1, keypoints1, None, flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
img2_with_keypoints = cv2.drawKeypoints(img2, keypoints2, None, flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

# Display the keypoints
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.imshow(img1_with_keypoints, cmap='gray')
plt.title('Keypoints in Image 1')

plt.subplot(1, 2, 2)
plt.imshow(img2_with_keypoints, cmap='gray')
plt.title('Keypoints in Image 2')

plt.show()
```




```
# Initialize the BFMatcher
bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)

# Match descriptors
matches = bf.match(descriptors1, descriptors2)

# Sort matches based on their distance (lower distance is better)
matches = sorted(matches, key=lambda x: x.distance)
# Draw the first 50 matches
img_matches = cv2.drawMatches(img1, keypoints1, img2, keypoints2, matches[:50], None, flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)

# Display the matches
plt.figure(figsize=(15, 10))
plt.imshow(img_matches)
plt.title('Feature Matching between Images')
plt.show()
```