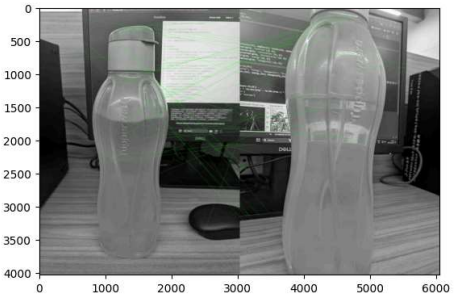
```
import numpy as np
import cv2
from google.colab.patches import cv2_imshow
bottle1=cv2.imread('/content/drive/MyDrive/PXL_20240229_093749300.jpg')
cv2_imshow(bottle1)
```


```
bottle2=cv2.imread('/content/drive/MyDrive/PXL_20240229_094805538.jpg')
cv2_imshow(bottle2)
```


```
import matplotlib.pyplot as plt
%matplotlib inline
img1 = cv2.cvtColor(bottle1, cv2.COLOR_BGR2GRAY)
img2 = cv2.cvtColor(bottle2, cv2.COLOR_BGR2GRAY)

#SIFT
sift =cv2.xfeatures2d.SIFT_create()
```

```
keypoints_1, descriptors_1 = sift.detectAndCompute(img1, None)
keypoints_2, descriptors_2 = sift.detectAndCompute(img2, None)


#feature matching
bf=cv2.BFMatcher(cv2.NORM_L1, crossCheck=True)

matches = bf.match(descriptors_1, descriptors_2)


matches = sorted(matches, key = lambda x:x.distance)


img3 = cv2.drawMatches(img1, keypoints_1, img2, keypoints_2, matches[:50], img2, flags = 2, matchColor=(0, 255, 0))
plt.imshow(img3), plt.show()
```



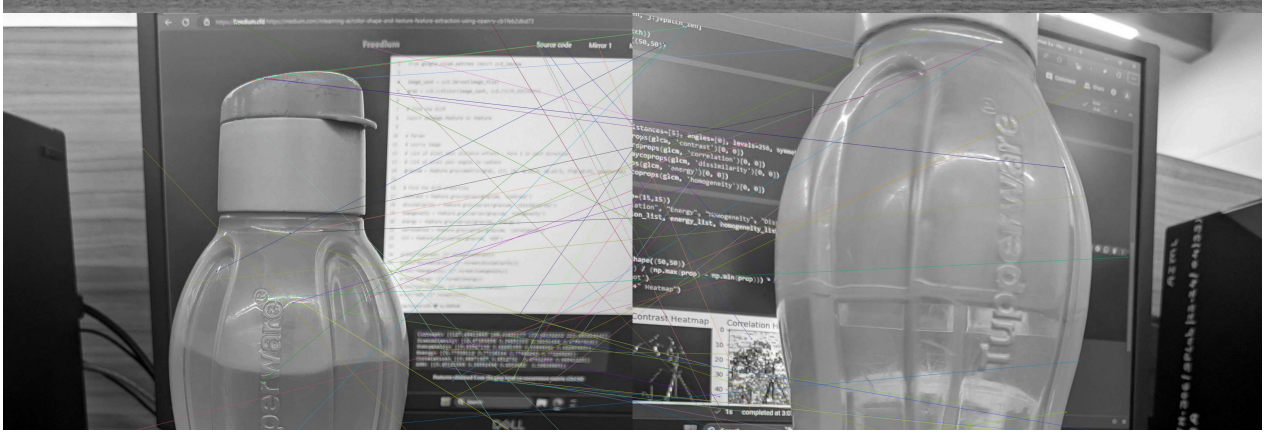

```
(<matplotlib.image.AxesImage at 0x7cdedb527af0>, None)
```

```
#extract matched keypoints
src_pts = np.float32([keypoints_1[m.queryIdx].pt for m in matches]).reshape(-1,1,2)
dst_pts = np.float32([keypoints_2[m.trainIdx].pt for m in matches]).reshape(-1,1,2)

#perform perspective transformation
M, _ = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC, 4.0)

#applying perspective transformations to first image
registered_image = cv2.warpPerspective(img2, M, (img1.shape[1], img1.shape[0]))

cv2_imshow(registered_image)


points_image1 = np.array([[x1, y1], [x2, y2], [x3, y3], [x4, y4]], dtype=np.float32)
points_image2 = np.array([[x1_, y1_], [x2_, y2_], [x3_, y3_], [x4_, y4_]], dtype=np.float32)

# Compute the perspective transform
matrix = cv2.getPerspectiveTransform(points_image2, points_image1)

# Apply the perspective transform to image2 to align it with image1
aligned_image2 = cv2.warpPerspective(img2, matrix, (img1.shape[1], img1.shape[0]))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-27-cd8cce47c711> in <cell line: 1>()
----> 1 points_image1 = np.array([[x1, y1], [x2, y2], [x3, y3], [x4, y4]], dtype=np.float32)
      2 points_image2 = np.array([[x1_, y1_], [x2_, y2_], [x3_, y3_], [x4_, y4_]], dtype=np.float32)
      3
      4 # Compute the perspective transform
      5 matrix = cv2.getPerspectiveTransform(points_image2, points_image1)

NameError: name 'x1' is not defined
```

```
# Check if the images are loaded successfully
if img1 is None or img2 is None:
    print("Error: Unable to load images.")
else:
    # Initialize the SIFT detector
    sift = cv2.SIFT_create()

    # Find keypoints and descriptors in both images
    keypoints_reference, descriptors_reference = sift.detectAndCompute(img1, None)
    keypoints_second, descriptors_second = sift.detectAndCompute(img2, None)

    # Match descriptors between the reference image and the second image
    matcher = cv2.BFMatcher()
    matches = matcher.knnMatch(descriptors_reference, descriptors_second, k=2)

    # Apply ratio test to find good matches
    good_matches = []
    for m, n in matches:
        if m.distance < 0.75 * n.distance:
            good_matches.append(m)

    # Extract keypoints from the good matches
    points_reference = np.float32([keypoints_reference[m.queryIdx].pt for m in good_matches]).reshape(-1, 1, 2)
    points_second = np.float32([keypoints_second[m.trainIdx].pt for m in good_matches]).reshape(-1, 1, 2)

    # Estimate the transformation (homography) between the two images
    homography, _ = cv2.findHomography(points_second, points_reference, cv2.RANSAC)

    # Apply the transformation to map the second image onto the reference image
    registered_image = cv2.warpPerspective(img2, homography, (img1.shape[1], img1.shape[0]))

    # Draw matching features on the concatenated image

    matching_image = cv2.drawMatches(img1, keypoints_reference, img2, keypoints_second, good_matches, None, flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)


    cv2_imshow(img1)
    cv2_imshow(matching_image)
```


Freedium
Source code
Mirror 1
Tupperware
DELL


Freedium
Source code
Mirror 1
Contrast Heatmap
Correlation Heatmap
Tupperware
DELL