Install Library

```
!apt-get update
!apt-get install -y cmake build-essential pkg-config

!git clone https://github.com/opencv/opencv.git
!git clone https://github.com/opencv/opencv_contrib.git

!mkdir -p opencv/build
%cd opencv/build
!cmake -D CMAKE_BUILD_TYPE=RELEASE \
        -D CMAKE_INSTALL_PREFIX=/usr/local \
        -D OPENCV_ENABLE_NONFREE=ON \
        -D OPENCV_EXTRA_MODULES_PATH=../../opencv_contrib/modules \
        -D BUILD_EXAMPLES=OFF ..
!make -j8
!make install
```

```
Hit:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:5 https://cloud.r-project.org/bin/linux/ubuntu jammy-cran40/ InRelease [3,626 B]
Hit:6 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86_64  InRelease
Hit:7 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu jammy InRelease
Get:8 https://ppa.launchpadcontent.net/graphics-drivers/ppa/ubuntu jammy InRelease [24.3 kB]
Hit:9 https://ppa.launchpadcontent.net/ubuntugis/ppa/ubuntu jammy InRelease
Ign:10 https://r2u.stat.illinois.edu/ubuntu jammy InRelease
Get:11 https://r2u.stat.illinois.edu/ubuntu jammy Release [5,713 B]
Get:12 https://r2u.stat.illinois.edu/ubuntu jammy Release.gpg [793 B]
Get:13 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [2,595 kB]
Get:14 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1,445 kB]
Get:15 http://archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [81.4 kB]
Get:16 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [33.7 kB]
Get:17 https://ppa.launchpadcontent.net/graphics-drivers/ppa/ubuntu jammy/main amd64 Packages [53.5 kB]
Get:18 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [1,156 kB]
Get:19 https://r2u.stat.illinois.edu/ubuntu jammy/main amd64 Packages [2,584 kB]
Get:20 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [2,318 kB]
Get:21 https://r2u.stat.illinois.edu/ubuntu jammy/main all Packages [8,352 kB]
Fetched 19.0 MB in 8s (2,521 kB/s)
Reading package lists... Done
W: Skipping acquire of configured file 'main/source/Sources' as repository 'https://r2u.stat.illinois.edu/ubu
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
build-essential is already the newest version (12.9ubuntu3).
cmake is already the newest version (3.22.1-1ubuntu1.22.04.2).
The following packages were automatically installed and are no longer required:
  libbz2-dev libpkgconf3 libreadline-dev
Use 'apt autoremove' to remove them.
The following packages will be REMOVED:
  pkgconf r-base-dev
The following NEW packages will be installed:
  pkg-config
0 upgraded, 1 newly installed, 2 to remove and 54 not upgraded.
Need to get 48.2 kB of archives.
After this operation, 11.3 kB disk space will be freed.
Get:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 pkg-config amd64 0.29.2-1ubuntu3 [48.2 kB]
Fetched 48.2 kB in 0s (144 kB/s)
(Reading database ... 123614 files and directories currently installed.)
Removing r-base-dev (4.4.1-3.2204.0) ...
dpkg: pkgconf: dependency problems, but removing anyway as you requested:
 libsndfile1-dev:amd64 depends on pkg-config; however:
  Package pkg-config is not installed.
  Package pkgconf which provides pkg-config is to be removed.
 libopencv-dev depends on pkg-config; however:
  Package pkg-config is not installed.
  Package pkgconf which provides pkg-config is to be removed.
 libmkl-dev:amd64 depends on pkg-config; however:
  Package pkg-config is not installed.
  Package pkgconf which provides pkg-config is to be removed.
 libjack-dev depends on pkg-config; however:
  Package pkg-config is not installed.
  Package pkgconf which provides pkg-config is to be removed.
 libgphoto2-dev:amd64 depends on pkg-config; however:
```

STEP 1 Load Two Images

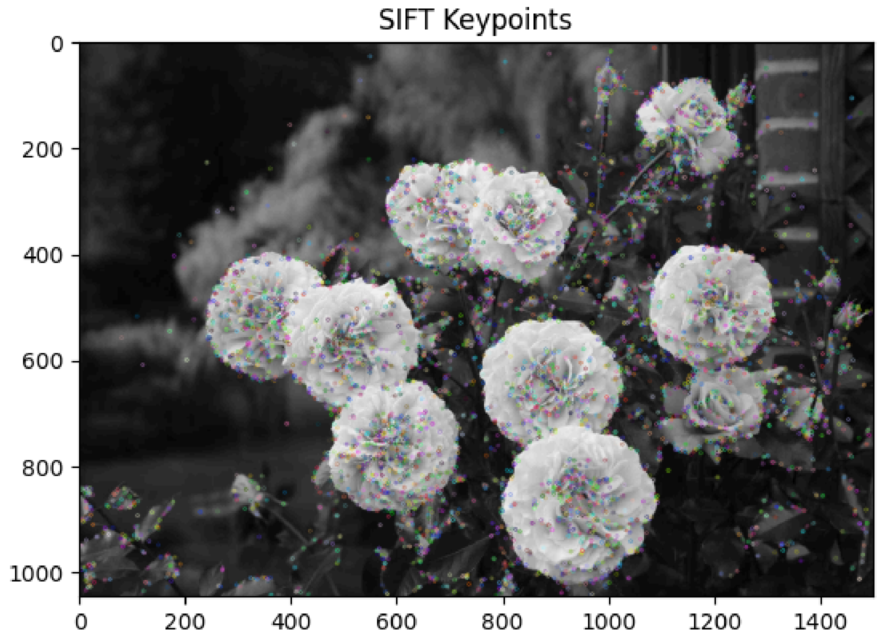
```
!pip install  opencv-python
!pip install  opencv-contrib-python
import cv2
import matplotlib.pyplot as plt
```

Show hidden output

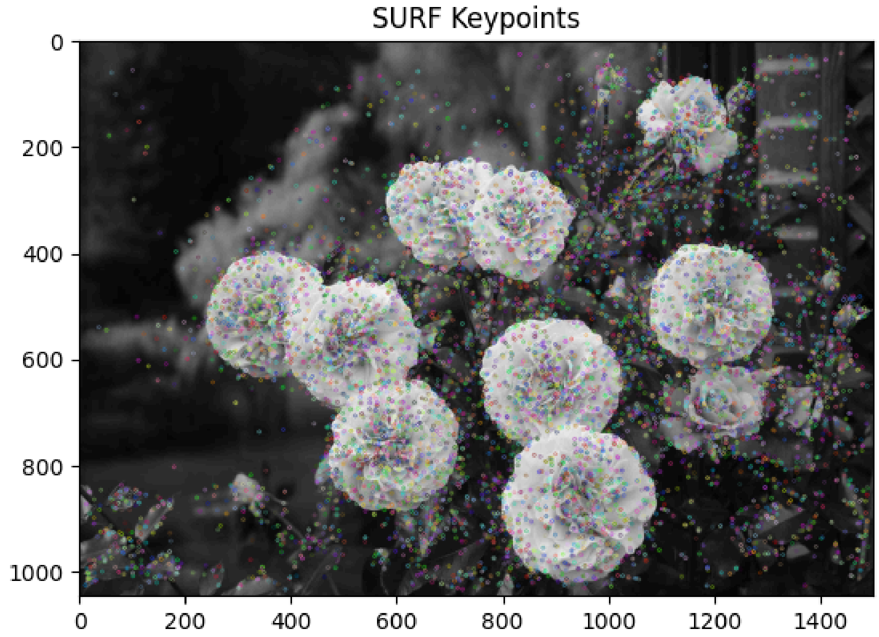
```
# Load two images
image1 = cv2.imread('/content/rose.jpg', 0)  # Replace with your actual image paths
image2 = cv2.imread('/content/daisy.jpg', 0)
```

Step 2: Extract Keypoints and Descriptors Using SIFT, SURF, and ORB

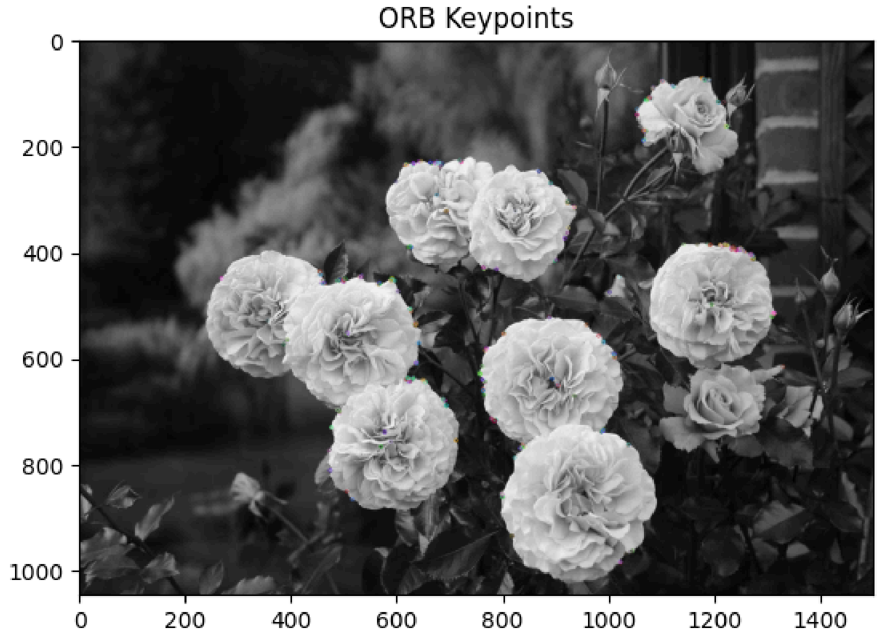
```
# Initialize SIFT
sift = cv2.SIFT_create()
keypoints1_sift, descriptors1_sift = sift.detectAndCompute(image1, None)
keypoints2_sift, descriptors2_sift = sift.detectAndCompute(image2, None)

# Visualize SIFT keypoints
sift_keypoints_image = cv2.drawKeypoints(image1, keypoints1_sift, None)
plt.imshow(cv2.cvtColor(sift_keypoints_image, cv2.COLOR_BGR2RGB))
plt.title('SIFT Keypoints')
plt.show()

# Initialize SURF (requires OpenCV contrib package)
surf = cv2.xfeatures2d.SURF_create()
keypoints1_surf, descriptors1_surf = surf.detectAndCompute(image1, None)
keypoints2_surf, descriptors2_surf = surf.detectAndCompute(image2, None)

# Visualize SURF keypoints
surf_keypoints_image = cv2.drawKeypoints(image1, keypoints1_surf, None)
plt.imshow(cv2.cvtColor(surf_keypoints_image, cv2.COLOR_BGR2RGB))
plt.title('SURF Keypoints')
plt.show()

# Initialize ORB
orb = cv2.ORB_create()
keypoints1_orb, descriptors1_orb = orb.detectAndCompute(image1, None)
keypoints2_orb, descriptors2_orb = orb.detectAndCompute(image2, None)

# Visualize ORB keypoints
orb_keypoints_image = cv2.drawKeypoints(image1, keypoints1_orb, None)
plt.imshow(cv2.cvtColor(orb_keypoints_image, cv2.COLOR_BGR2RGB))
plt.title('ORB Keypoints')
plt.show()


# Save keypoints visualization images
cv2.imwrite('sift_keypoints.jpg', cv2.drawKeypoints(image1, keypoints1_sift, None))
cv2.imwrite('surf_keypoints.jpg', cv2.drawKeypoints(image1, keypoints1_surf, None))
cv2.imwrite('orb_keypoints.jpg', cv2.drawKeypoints(image1, keypoints1_orb, None))
```



SIFT Keypoints



SURF Keypoints



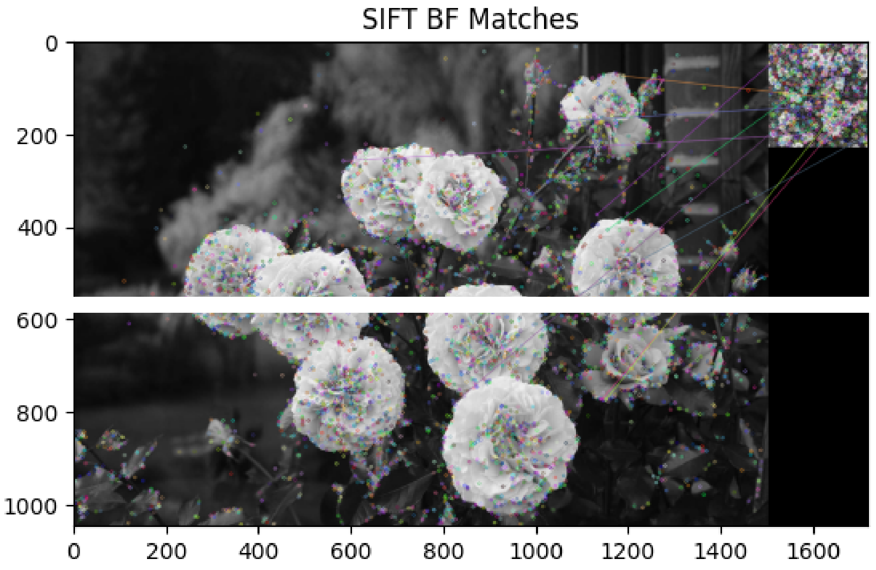
ORB Keypoints

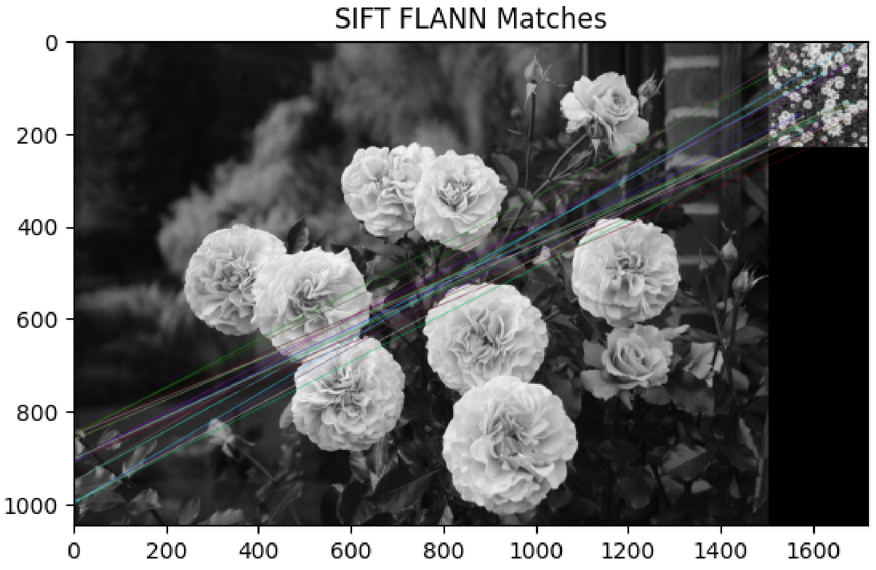
```
True
```

Step 3: Feature Matching with Brute-Force and FLANN

```
# Brute-Force Matcher for SIFT
bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)
matches_sift_bf = bf.match(descriptors1_sift, descriptors2_sift)
matches_sift_bf = sorted(matches_sift_bf, key=lambda x: x.distance)

# Visualize Brute-Force SIFT matches
sift_bf_img = cv2.drawMatches(image1, keypoints1_sift, image2, keypoints2_sift, matches_sift_bf[:10], None)
plt.imshow(cv2.cvtColor(sift_bf_img, cv2.COLOR_BGR2RGB))
plt.title('SIFT BF Matches')
plt.show()
```

```
# FLANN Matcher for SIFT
flann_index_kdtree = 1
index_params = dict(algorithm=flann_index_kdtree, trees=5)
search_params = dict(checks=50)  # or pass empty dictionary

flann = cv2.FlannBasedMatcher(index_params, search_params)
matches_sift_flann = flann.knnMatch(descriptors1_sift, descriptors2_sift, k=2)

# Visualize FLANN matches
sift_flann_img = cv2.drawMatchesKnn(image1, keypoints1_sift, image2, keypoints2_sift, matches_sift_flann[:10], None,
plt.imshow(cv2.cvtColor(sift_flann_img, cv2.COLOR_BGR2RGB))
plt.title('SIFT FLANN Matches')
plt.show()

# Drawing matches
sift_bf_img = cv2.drawMatches(image1, keypoints1_sift, image2, keypoints2_sift, matches_sift_bf[:10], None)
cv2.imwrite('sift_bf_match.jpg', sift_bf_img)

sift_flann_img = cv2.drawMatchesKnn(image1, keypoints1_sift, image2, keypoints2_sift, matches_sift_flann[:10], None,
cv2.imwrite('sift_flann_match.jpg', sift_flann_img)
```





```
True
```

Step 4: Image Alignment Using Homography

```
import numpy as np

# Use the matched keypoints to compute the homography matrix
src_pts = np.float32([keypoints1_sift[m.queryIdx].pt for m in matches_sift_bf]).reshape(-1, 1, 2)
dst_pts = np.float32([keypoints2_sift[m.trainIdx].pt for m in matches_sift_bf]).reshape(-1, 1, 2)

# Compute the homography matrix
H, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC, 5.0)

# Warp image1 to align with image2
aligned_image = cv2.warpPerspective(image1, H, (image2.shape[1], image2.shape[0]))
cv2.imwrite('aligned_image.jpg', aligned_image)

# Visualize aligned image
plt.imshow(cv2.cvtColor(aligned_image, cv2.COLOR_BGR2RGB))
plt.title('Aligned Image')
plt.show()
```