Installing Packages

```
!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
```

```
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/fluid/gfluidbuffer.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/fluid/gfluidkernel.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/fluid/imgproc.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/garg.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/garray.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/gasync_context.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/gcall.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/gcommon.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/gcompiled.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/gcompiled_async.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/gcompoundkernel.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/gcomputation.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/gcomputation_async.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/gframe.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/gkernel.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/gmat.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/gmetaarg.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/gopaque.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/gproto.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/gpu/core.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/gpu/ggpukernel.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/gpu/imgproc.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/gscalar.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/gstreaming.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/gtransform.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/gtype_traits.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/gtyped.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/imgproc.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/infer.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/infer/bindings_ie.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/infer/bindings_onnx.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/infer/bindings_ov.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/infer/ie.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/infer/onnx.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/infer/ov.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/infer/parsers.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/media.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/oak/infer.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/oak/oak.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/ocl/core.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/ocl/goclkernel.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/ocl/imgproc.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/opencv_includes.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/operators.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/ot.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/own/assert.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/own/convert.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/own/cvdefs.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/own/exports.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/own/mat.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/own/saturate.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/own/scalar.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/own/types.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/plaidml/core.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/plaidml/gplaidmlkernel.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/plaidml/plaidml.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/python/python.hpp
-- Up-to-date: /usr/local/include/opencv4/opencv2/gapi/render.hpp
```

STEP 1 Load Two Images

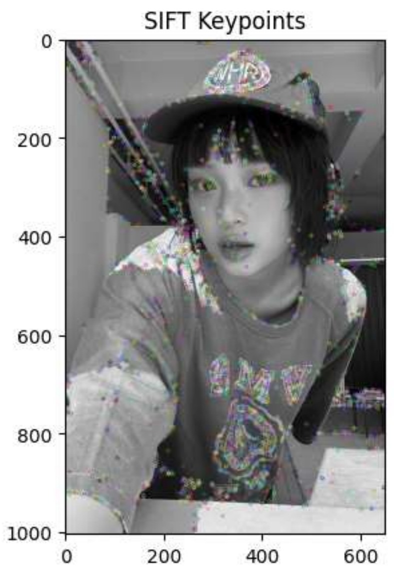
```
!pip install  opencv-python
!pip install  opencv-contrib-python
import matplotlib.pyplot as plt
import cv2
```

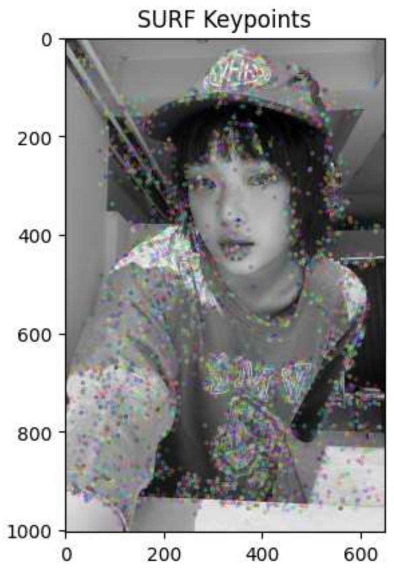
```
Requirement already satisfied: opencv-python in /usr/local/lib/python3.10/dist-packages (4.10.0.84)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from opencv-python) (1.26.4)
Requirement already satisfied: opencv-contrib-python in /usr/local/lib/python3.10/dist-packages (4.10.0.84)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from opencv-contrib-python) (1.26.4)
```

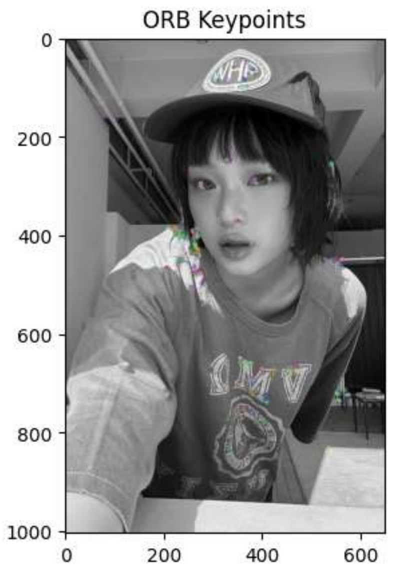
```
# Load two images
image1 = cv2.imread('/content/hanni.jpg', 0)  # Replace with your actual image paths
image2 = cv2.imread('/content/hanni2.jpg', 0)
```

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))
```







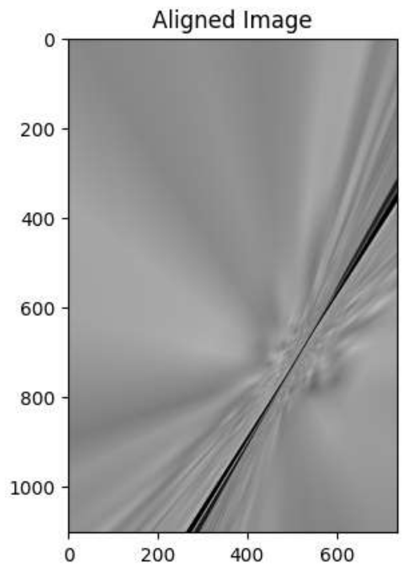
```
True
```

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, flags=cv2.DrawMatchesFl
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, flags=cv2.DrawMatchesFl
cv2.imwrite('sift_flann_match.jpg', sift_flann_img)
```

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()
```


Aligned Image