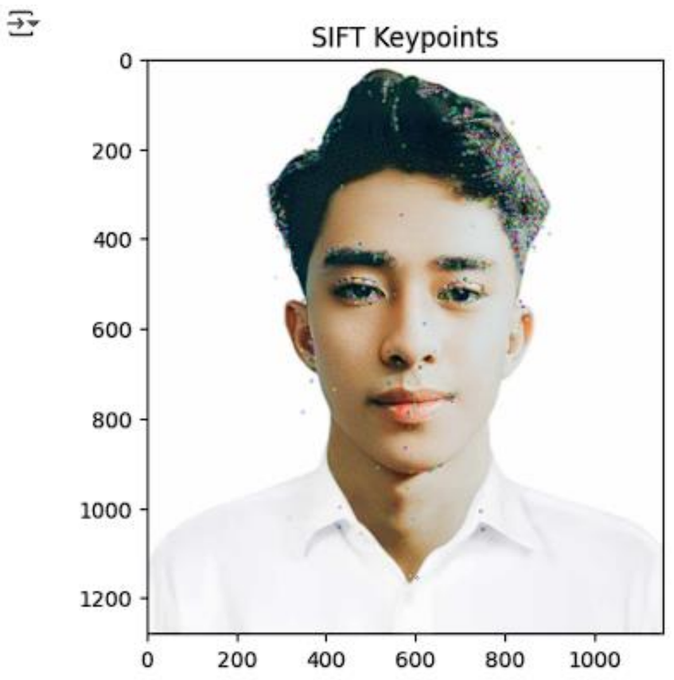
Feature Extraction and Object Detection

Task 1: SIFT Feature Extraction

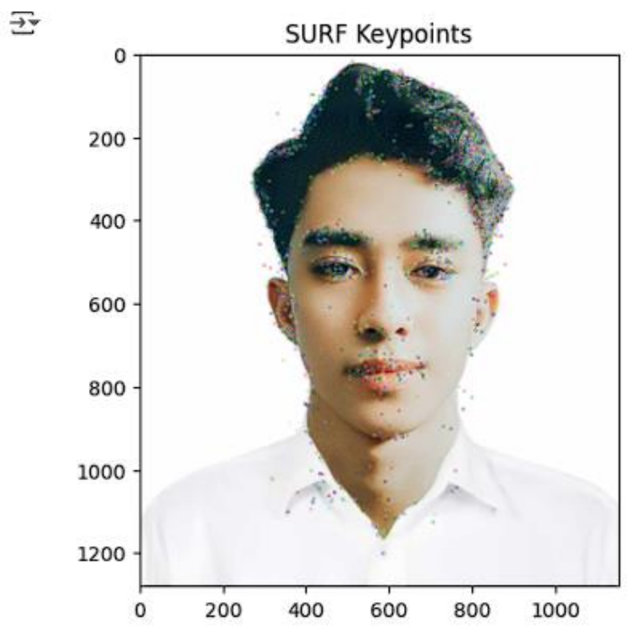
Output:



In this part I used SIFT for feature extraction on my image. I began by loading the image and converting it to grayscale. After initializing the SIFT detector, I detected keypoints and computed their descriptors. The keypoints represent important features in the image. Finally, I drew the detected keypoints on the original image and displayed it using Matplotlib. The resulting image clearly shows various points of interest, indicating the regions that SIFT identified significant features.

Task 2: SURF Feature Extraction

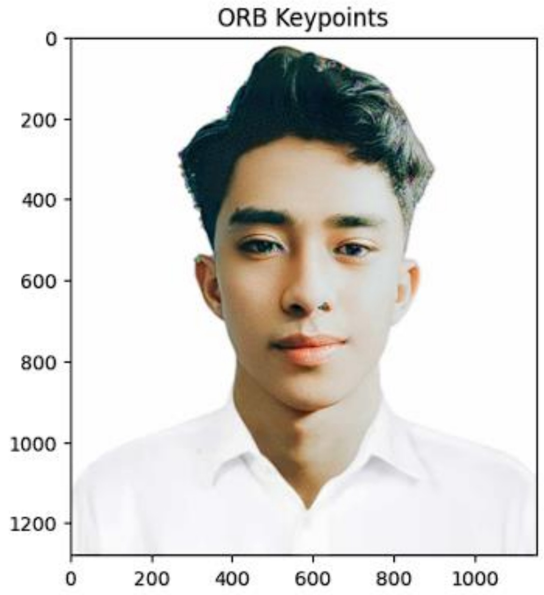
Output:



In this task, I implemented feature extraction using the Speeded-Up Robust Features (SURF) algorithm. I began by loading the image and converting it to grayscale, which is necessary for SURF processing. After initializing the SURF detector, I detected keypoints and computed their descriptors, which represent important features in the image. The keypoints were drawn on the original image to visualize the significant areas identified by the algorithm. The resulting image clearly showed various keypoints, indicating the distinct features that SURF successfully detected.

Task 3: ORB Feature Extraction

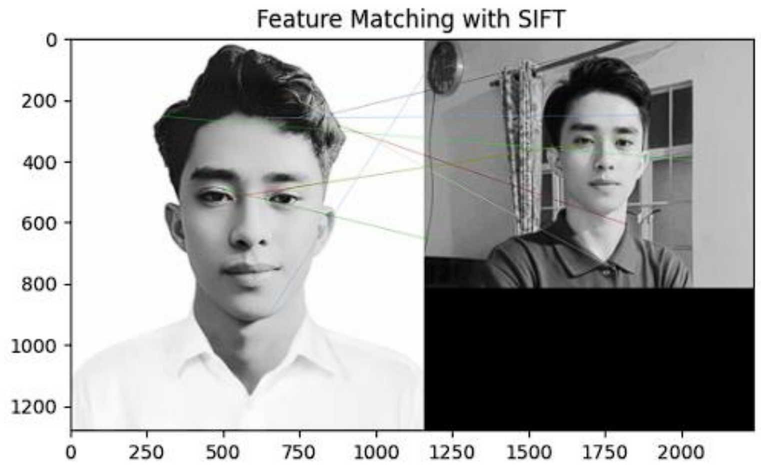
Output:



In this task, I used ORB algorithm to extract features from my image. I started by loading the image and converting it to grayscale, which is essential for feature detection. Next, I initialized the ORB detector and proceeded to detect keypoints and compute their descriptors, which encapsulate the distinctive characteristics of the image. After extracting these features, I visualized the results by drawing the keypoints on the original image. The displayed output highlighted various keypoints, demonstrating the effective identification of significant features by the ORB algorithm.

Task 4: Feature Matching

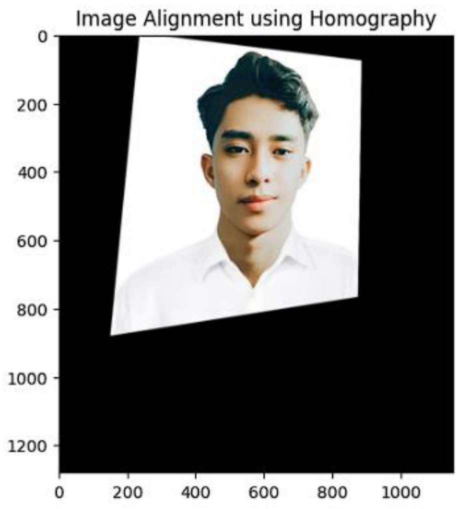
Output:



I performed feature matching between two images using the Speeded-Up Robust Features (SIFT) algorithm. I began by loading both images in grayscale to facilitate the SIFT processing. After initializing the SIFT detector, I detected keypoints and computed their descriptors for both images. To match the features between the two images, I used a brute-force matcher with L2 norm and cross-checking for accuracy. The matched descriptors were sorted by distance to prioritize the best matches. Finally, I visualized the results by drawing the top matches on a combined image, which effectively illustrated how SIFT identified and linked corresponding features between the two images.

Task 5: Application of Feature Matching

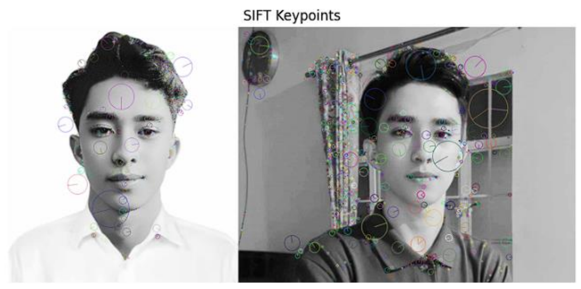
Output:

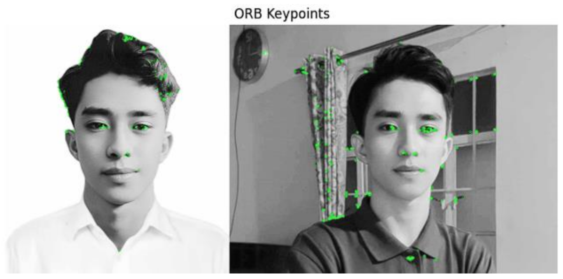


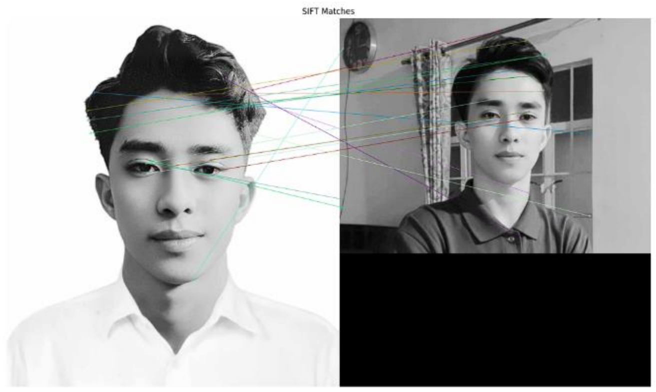
In this task, I focused on aligning two images based on feature matching using the SIFT algorithm. I started by loading both images and converting them to grayscale to facilitate the processing. After detecting keypoints and computing descriptors for both images using SIFT, I employed the Brute Force Matcher to find matches between the features. To enhance the matching quality, I applied Lowe's ratio test, which filtered out weaker matches by ensuring that the best match was significantly better than the second-best match. Next, I extracted the coordinates of the good matches and used them to calculate the homography matrix, which defines the transformation needed to align the two images. Using the computed homography matrix, I warped the first image to align it with the second image. Finally, I displayed the aligned image, demonstrating how effectively SIFT and homography can be used to merge different perspectives of the same scene.

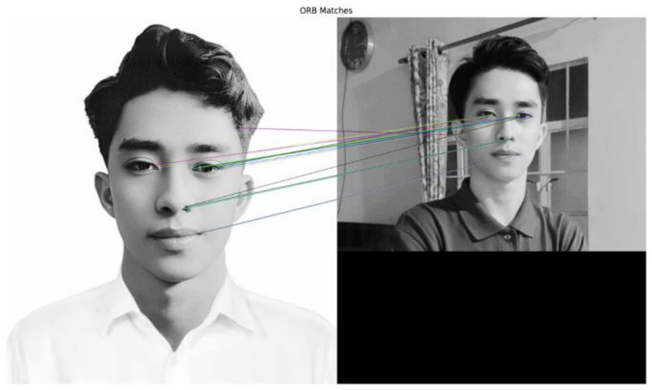
Task 6: Combining Feature Extraction Method

Output:









In the last task, I performed feature extraction and matching on two images using SIFT and ORB algorithms to evaluate their effectiveness. First, I loaded the two images and applied the SIFT algorithm to detect keypoints and compute descriptors, capturing essential features of the images. Similarly, I utilized the ORB detector to extract features from the same images. To visualize the keypoints detected by both methods, I drew the keypoints on the original images and resized them for better comparison, resulting in combined images that showcase the detected features side by side.

After visualizing the keypoints, I implemented a function to match the descriptors obtained from both SIFT and ORB. Using the Brute Force Matcher, I matched the descriptors and sorted them by distance to highlight the strongest matches. I displayed the top 20 matches for both algorithms, providing insight into their matching capabilities. The results illustrate the effectiveness of SIFT and ORB in feature extraction and matching, demonstrating the utility of these techniques in image analysis and computer vision tasks.