Documentation for Exercises No. 2: Feature Extraction and Object Detection

1. Introduction

This notebook demonstrates the process of feature extraction and object detection using different feature detection methods, such as SIFT (Scale-Invariant Feature Transform), SURF (Speeded-Up Robust Features), and ORB (Oriented FAST and Rotated BRIEF). The steps include loading images, extracting features, matching features between images, and aligning images based on feature matching.

Key Tasks:

1. **SIFT Feature Extraction**
2. **SURF Feature Extraction**
3. **ORB Feature Extraction**
4. **Feature Matching**
5. **Applications of Feature Matching (Homography and Image Alignment)**
6. **Combining Different Feature Extraction Methods**

2. Importing Necessary Libraries

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab import files
```

These libraries are required for image processing (using OpenCV), numerical operations (using NumPy), and visualization (using Matplotlib).

3. Loading Images

```
image1 = cv2.imread('/iMAGE2.jpg')
image2 = cv2.imread('/IMAGE3.jpg')
```

We load two images (`image1` and `image2`) from the file system. The images are then verified for successful loading.

Verification:

```
if image1 is None:
    print("Error loading image1!")
else:
    print(f"Image1 loaded with shape: {image1.shape}")
```

```
if image2 is None:
    print("Error loading image2!")
else:
    print(f"Image2 loaded with shape: {image2.shape}")
```

This code ensures that both images are loaded correctly. If an image fails to load, it prints an error message.

4. Converting Images to Grayscale

```
gray1 = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)
gray2 = cv2.cvtColor(image2, cv2.COLOR_BGR2GRAY)
```

Both images are converted to grayscale to simplify feature detection. This step is necessary because feature extraction algorithms generally work with grayscale images.

5. Task 1: SIFT Feature Extraction

```
sift = cv2.SIFT_create()
kp1_sift, des1_sift = sift.detectAndCompute(gray1, None)
kp2_sift, des2_sift = sift.detectAndCompute(gray2, None)
```

SIFT (Scale-Invariant Feature Transform) is used to detect keypoints and compute descriptors for the images.

Displaying the Keypoints:

```
image_matches_sift = cv2.drawMatches(image1, kp1_sift, image2, kp2_sift,
None, None, flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
plt.imshow(cv2.cvtColor(image_matches_sift, cv2.COLOR_BGR2RGB))
plt.title("SIFT Feature Extraction")
plt.axis('off')
plt.show()
cv2.imwrite('/content/4B-BERCADES-EXER2-sift-keypoints.jpg',
image_matches_sift)
```

This code visualizes the keypoints detected by SIFT and saves the image for future use.

6. Task 2: SURF Feature Extraction

SURF (Speeded-Up Robust Features) is another feature detection algorithm, but it is non-free and may not be available in all OpenCV installations. We attempt to initialize the SURF detector:

```
try:
    surf = cv2.xfeatures2d.SURF_create()
    print("SURF is ready to use!")
except:
```

```
    print("SURF is still not available.")
```

If SURF is available, we proceed with feature extraction:

```
keypoints, descriptors = surf.detectAndCompute(gray1, None)
image_with_keypoints = cv2.drawKeypoints(image1, keypoints, None,
flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
plt.imshow(cv2.cvtColor(image_with_keypoints, cv2.COLOR_BGR2RGB))
plt.title("SURF Keypoints")
plt.axis('off')
plt.show()
```

Note:

SURF requires non-free modules, which may be disabled in certain OpenCV builds.

7. Task 3: ORB Feature Extraction

ORB (Oriented FAST and Rotated BRIEF) is a fast and efficient feature detector and descriptor.

```
orb = cv2.ORB_create()
kp1_orb, des1_orb = orb.detectAndCompute(gray1, None)
kp2_orb, des2_orb = orb.detectAndCompute(gray2, None)
```

ORB keypoints are detected and displayed as follows:

```
image_matches_orb = cv2.drawMatches(image1, kp1_orb, image2, kp2_orb, None,
None, flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
plt.imshow(cv2.cvtColor(image_matches_orb, cv2.COLOR_BGR2RGB))
plt.title("ORB Feature Extraction")
plt.axis('off')
plt.show()
cv2.imwrite('/content/4B-BERCADES-EXER2-orb-keypoints.jpg',
image_matches_orb)
```

8. Task 4: Feature Matching

SIFT Matching

```
bf_sift = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)
matches_sift = bf_sift.match(des1_sift, des2_sift)
matches_sift = sorted(matches_sift, key=lambda x: x.distance)
```

ORB Matching

```
bf_orb = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
matches_orb = bf_orb.match(des1_orb, des2_orb)
matches_orb = sorted(matches_orb, key=lambda x: x.distance)
```

We use the brute-force matcher for both SIFT and ORB descriptors and sort the matches by distance to find the best ones.

Displaying the First 10 Matches for SIFT and ORB

```
image_matches_sift_combined = cv2.drawMatches(image1, kp1_sift, image2,
kp2_sift, matches_sift[:10], None,
flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
image_matches_orb_combined = cv2.drawMatches(image1, kp1_orb, image2,
kp2_orb, matches_orb[:10], None,
flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
```

We display the top 10 matches for both SIFT and ORB in side-by-side images.

9. Task 5: Applications of Feature Matching

Homography and Image Alignment Using ORB Matches

```
pts1_orb = np.float32([kp1_orb[m.queryIdx].pt for m in
matches_orb]).reshape(-1, 1, 2)
pts2_orb = np.float32([kp2_orb[m.trainIdx].pt for m in
matches_orb]).reshape(-1, 1, 2)

H_orb, mask_orb = cv2.findHomography(pts1_orb, pts2_orb, cv2.RANSAC, 5.0)
aligned_image_orb = cv2.warpPerspective(image1, H_orb, (image2.shape[1],
image2.shape[0]))
```

This code uses RANSAC to compute the homography matrix and then warps the first image to align with the second image.

Displaying the Aligned Image

```
plt.imshow(cv2.cvtColor(aligned_image_orb, cv2.COLOR_BGR2RGB))
plt.title("Aligned Image using Homography (ORB)")
plt.axis('off')
plt.show()
```

10. Task 6: Combining Feature Extraction Methods

In this task, SIFT and ORB feature extraction methods are applied, and their matches are displayed and aligned using homography.

```
# Repeat SIFT and ORB matching and homography steps
```

We visualize the results of SIFT and ORB matches and aligned images for further analysis.

11. Conclusion

In this exercise, we explored three feature extraction methods: **SIFT**, **SURF**, and **ORB**, applied feature matching, and demonstrated practical applications such as image alignment through homography. The methods are useful for image stitching, object recognition, and other computer vision tasks.

12. Image Download

The results are saved and made available for download:

```
cv2.imwrite('/content/4B-BERCADES-EXER2-sift-matches.jpg',
image_matches_sift_combined)
cv2.imwrite('/content/4B-BERCADES-EXER2-orb-matches.jpg',
image_matches_orb_combined)
cv2.imwrite('/content/4B-BERCADES-EXER2-aligned-orb.jpg', aligned_image_orb)
files.download('/content/4B-BERCADES-EXER2-sift-matches.jpg')
files.download('/content/4B-BERCADES-EXER2-orb-matches.jpg')
files.download('/content/4B-BERCADES-EXER2-aligned-orb.jpg')
```

This code allows users to download the output images of matches and aligned images after processing.

This documentation provides a step-by-step guide through the feature extraction, matching, and application processes in object detection using OpenCV.