COURSE CODE: CSE4047
COURSE NAME : COMPUTER VISION

Name: Syed. Mahammed Sameer
RegNo: 21bce8463
Date: 04-11-2024

Google Drive : Link

PROJECT-4

TITLE-

Image Stitching

Steps:

· · **Image Acquisition:**

- Load two images from specified local paths.
- Ensure both images are present; otherwise, print a message indicating insufficient images for stitching.

· **Feature Detection:**

- Convert both images to grayscale to prepare them for feature detection (grayscale images simplify the process by reducing data dimensions).
- Initialize a SIFT (Scale-Invariant Feature Transform) detector, which is used to identify keypoints and extract distinctive features from the images.
- Detect and compute the SIFT keypoints and descriptors for each grayscale image. Store these keypoints and descriptors for further use in matching.

· **Feature Matching:**

- Use BFMatcher (Brute Force Matcher) to find matches between the descriptors of the two images. This matcher looks for similar descriptors between the two sets.
- Use cross-checking to filter out weak matches, ensuring each matched feature is mutual.
- Sort the matches based on distance to retain the best matches (closest matches have the smallest distance).
- For visualization, display the top 50 matches between the two images, which helps verify that the features were detected and matched correctly.

· **Homography Estimation:**

- Extract the matched points from both images using the indices provided by the matches.
- Calculate the homography matrix using RANSAC (Random Sample Consensus) to filter out any mismatched points. The homography matrix defines the transformation needed to align one image with the other.

· **Image Warping and Alignment:**

- Warp the first image using the calculated homography matrix so that it aligns with the second image.
- Adjust the width of the canvas for the warped image to accommodate both images without clipping.
- Place the second image on the warped result to begin forming a stitched image.

· **Image Blending:**

- Optionally, apply blending techniques to smooth the seam where the images meet. This step would help make the stitched image appear more seamless.

· **Rendering:**

- Display the final stitched image using Matplotlib for a clear view of the result.
- Turn off the axis for a cleaner presentation.

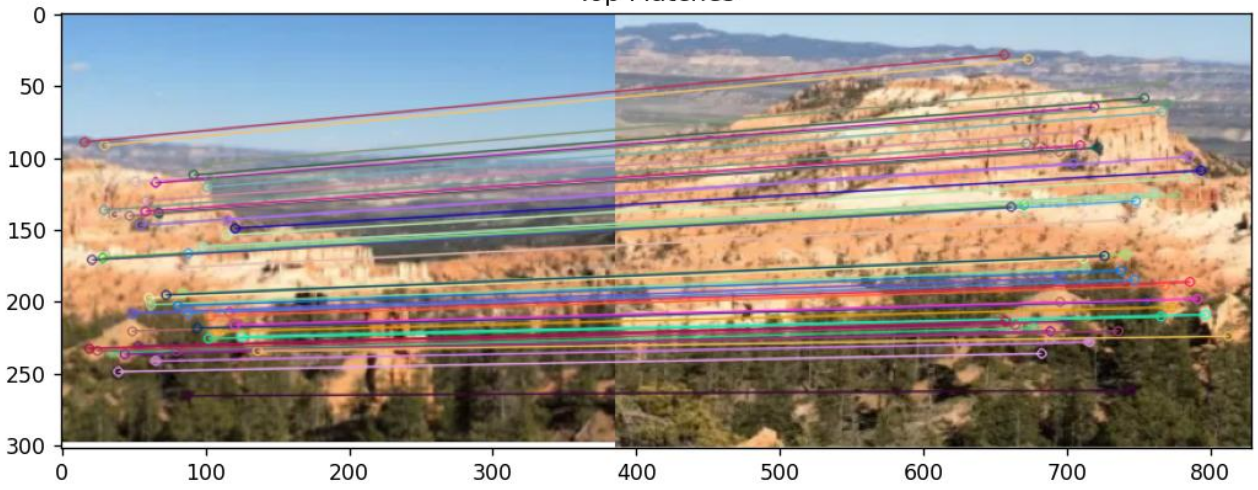
Code:

```
import cv2
import numpy as np
from skimage import io
import matplotlib.pyplot as plt

# 1) Image Acquisition:
# Load local images (Make sure to replace these with your actual file paths)
image_paths = [
    "C:/Users/Sameer/Downloads/Screenshot 2024-11-07 224311.png",  # Replace with your
local file path
    "C:/Users/Sameer/Downloads/Screenshot 2024-11-07 224252.png"   # Replace with your
local file path
]

# Read images from local paths
images = [io.imread(image_path) for image_path in image_paths]

if len(images) < 2:
    print("Not enough images to proceed with stitching.")
else:
    # 2) Feature Detection:
    # Convert images to grayscale
    gray_images = [cv2.cvtColor(img, cv2.COLOR_RGB2GRAY) for img in images]

    # Initialize SIFT detector
    sift = cv2.SIFT_create()
```

```
    # Detect and compute features
    keypoints, descriptors = [], []
    for gray_img in gray_images:
        kp, des = sift.detectAndCompute(gray_img, None)
        keypoints.append(kp)
        descriptors.append(des)

    # 3) Feature Matching:
    # Use BFMatcher to find feature matches
    bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)
    matches = bf.match(descriptors[0], descriptors[1])
    matches = sorted(matches, key=lambda x: x.distance)

    # Draw matches (for visualization purposes)
    matched_img = cv2.drawMatches(images[0], keypoints[0], images[1], keypoints[1],
matches[:50], None, flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
    plt.figure(figsize=(10, 5))
    plt.imshow(matched_img)
    plt.title("Top Matches")
    plt.show()

    # 4) Homography estimation:
    # Extract points for homography calculation
    src_pts = np.float32([keypoints[0][m.queryIdx].pt for m in matches]).reshape(-1, 1,
2)
    dst_pts = np.float32([keypoints[1][m.trainIdx].pt for m in matches]).reshape(-1, 1,
2)

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

    # 5) Image warping and alignment:
    # Warp the first image to align with the second
    height, width = images[1].shape[:2]
    warped_image = cv2.warpPerspective(images[0], H, (width + images[0].shape[1],
height))

    # Place the second image on the warped result for stitching
    warped_image[0:height, 0:width] = images[1]

    # 6) Image Blending:
    # Here we can add blending logic if needed to make the seams less visible

    # 7) Rendering:
    # Display the result
    plt.figure(figsize=(10, 5))
    plt.imshow(warped_image)
    plt.title("Stitched Image")
    plt.axis("off")
    plt.show()
```

Output:

Top Matches



Stitched Image