Project – Report

Omkar. A. Chittar

119193556

Q. You are given four images which were taken from the same camera position (only rotation no translation) you will need to stitch these images to create a panoramic image.
To solve this problem, you will need to:

- **Extract features from each frame (You can use any feature extractor).**
- **Match the features between each consecutive image and visualize them.**
- **Compute the homographies between the pairs of images.**
- **Combine these frames together using the computed homographies.**

Approach:

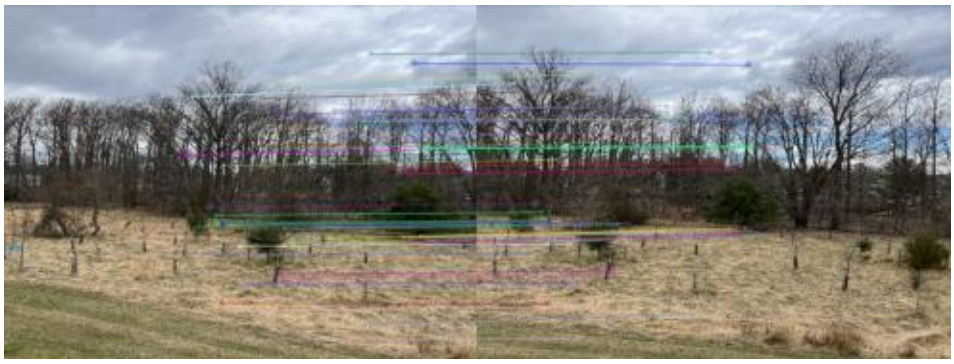
1. Import the req libraries - **cv2**, **numpy**, **scipy**, and **matplotlib**.
2. Define an empty list to store the images.
3. Define a function to perform RANSAC algorithm for homography calculation.
4. Define a function to compute homography given a set of source and destination keypoints.
5. Define a function to normalize the keypoints.
6. In the RANSAC function, normalize the source and destination keypoints.
7. Loop for the given number of iterations:
    a. Randomly select four pairs of matched keypoints.
    b. Calculate homography using these four pairs of keypoints.
    c. Apply the calculated homography to the source keypoints to obtain the predicted destination keypoints.
    d. Compute the distance between predicted and actual destination keypoints.
    e. Count the number of inliers based on a given threshold distance.
    f. Update the best homography if the current homography has more inliers than the previous homography or if it has the same number of inliers but with a smaller sum of distances.
8. Denormalize the final homography matrix.
9. Define a function to extract good matches between two images using BFMatcher.
10. Loop through all the images and perform registration between consecutive pairs of images using the above-defined functions.
11. Use the SIFT function to detect the features and use BFMatcher to match these features in adjacent images as shown in the results.
12. Use the homography matrix to warp two images at a time and create a panoramic image.

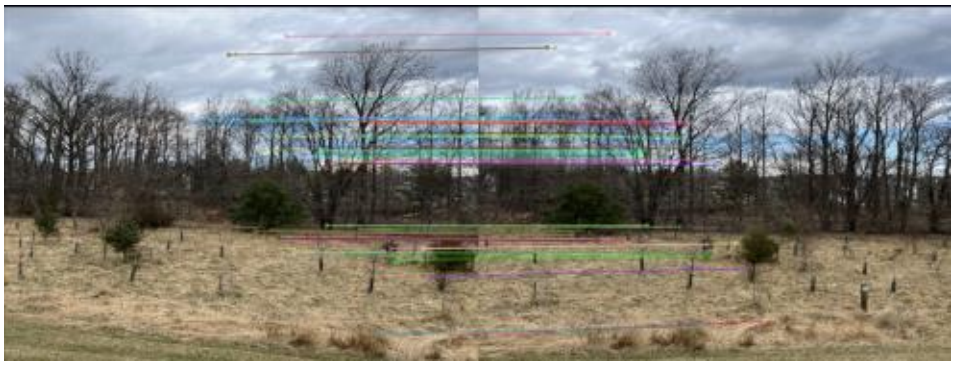
Results:

```
H12
[[ 1.30319370e+00  1.15747455e-01  4.68366582e+02]
 [ 7.27766043e-02  1.01825477e+00 -5.12912428e+01]
 [ 2.70807980e-04  8.22448055e-05  8.41188437e-01]]

H23
[[ 4.72011314e-01 -5.95969112e-02  1.19411592e+03]
 [-1.20011719e-01  9.72164368e-01  4.53641476e+01]
 [-3.86358173e-04 -4.46714849e-05  1.10594895e+00]]

H34
[[ 1.55621495e-02  8.16554785e-02  1.45484381e+03]
 [-1.96755781e-01  1.03191774e+00  2.79153773e+01]
 [-6.84471618e-04  6.00828349e-05  1.08669753e+00]]
```





Problems Encountered:

- Figuring out how many matches to consider for matching took a lot of time.
- The given image was too large for the **bfmatcher** to handle.