**Implementation of image registration method to find transform between the two image**

**Introduction**

Image registration is the process of aligning two or more images of the same scene into a common coordinate system. It is widely used in medical imaging, remote sensing, and computer vision applications.

In this work, we align a **distorted/rotated astronaut image** (moving image) to a **reference astronaut image** using a **feature-based method**.

**Theory of Image Registration**

**2.1 Feature-based Registration**

Feature-based registration relies on detecting keypoints (corners, edges, blobs) in both images and describing them using descriptors. By matching corresponding features between the reference and input image, we can compute a geometric transformation that aligns them.

**2.2 Homography Transform**

The geometric relationship between the input (moving) image and the reference image can be represented using a **homography matrix**:

A homography can correct for:

* Translation
* Rotation
* Scaling
* Perspective distortions

In practice, HHH is estimated using **RANSAC (Random Sample Consensus)** to remove outliers in feature matching.

**Methodology**

1. **Convert images to grayscale**.
2. **Extract features** using SIFT (Scale Invariant Feature Transform).
3. **Match descriptors** between images using the FLANN-based matcher.
4. **Filter matches** using Lowe’s ratio test.
5. **Estimate homography matrix** HHH with RANSAC.
6. **Warp the moving image** using cv2.warpPerspective to align with the reference.

**Implementation (Python Code)**

import cv2

import numpy as np

import matplotlib.pyplot as plt

def register\_images(ref\_img\_path, input\_img\_path):

# Load grayscale images

ref\_img = cv2.imread(ref\_img\_path, cv2.IMREAD\_GRAYSCALE)

input\_img = cv2.imread(input\_img\_path, cv2.IMREAD\_GRAYSCALE)

# Step 1: Detect and compute SIFT features

sift = cv2.SIFT\_create()

kp1, des1 = sift.detectAndCompute(ref\_img, None)

kp2, des2 = sift.detectAndCompute(input\_img, None)

# Step 2: Match features using FLANN

index\_params = dict(algorithm=1, trees=5)

search\_params = dict(checks=50)

flann = cv2.FlannBasedMatcher(index\_params, search\_params)

matches = flann.knnMatch(des1, des2, k=2)

# Step 3: Lowe's ratio test

good = []

for m, n in matches:

if m.distance < 0.7 \* n.distance:

good.append(m)

# Step 4: Compute homography

pts1 = np.float32([kp1[m.queryIdx].pt for m in good]).reshape(-1, 1, 2)

pts2 = np.float32([kp2[m.trainIdx].pt for m in good]).reshape(-1, 1, 2)

H, mask = cv2.findHomography(pts2, pts1, cv2.RANSAC)

# Step 5: Warp moving image

h, w = ref\_img.shape

aligned\_img = cv2.warpPerspective(input\_img, H, (w, h))

# Step 6: Display results

plt.figure(figsize=(15,5))

plt.subplot(1,3,1), plt.title("Reference Image"), plt.imshow(ref\_img, cmap="gray")

plt.subplot(1,3,2), plt.title("Input (Moving Image)"), plt.imshow(input\_img, cmap="gray")

plt.subplot(1,3,3), plt.title("Aligned Image"), plt.imshow(aligned\_img, cmap="gray")

plt.show()

cv2.imwrite("aligned\_output.png", aligned\_img)

print("✅ Aligned image saved as 'aligned\_output.png'")

return aligned\_img, H

aligned, H = register\_images("reference.png", "moving.png")

**Results**

**Input Images:**

* **Reference image (reference.png)** – Upright astronaut photograph.
* **Moving image (moving.png)** – Rotated and noisy astronaut photograph.

**Output:**

* The moving image was successfully registered to the reference image.
* The resulting aligned image is saved as **aligned\_output.png**.

The estimated **homography matrix** was also obtained:

This transformation corrects for rotation, translation, scaling, and perspective distortion.