Image Stitching

Ajitesh Kumar Singh

February 17, 2025

Abstract

This report details the implementation of an image stitching pipeline using feature detection, feature matching, and homography estimation. The system employs SIFT for keypoint detection, BFMatcher for feature matching, RANSAC for outlier rejection, and image blending for seamless stitching. The decisions made at each step are discussed, along with alternative approaches and their potential trade-offs, focusing on OpenCV functions used and their impact.

1 Introduction

Image stitching is a technique used to combine multiple images into a single panoramic image. It is widely used in computer vision applications such as panoramic photography and 3D reconstruction.

This project implements an automated image stitching pipeline using OpenCV. The steps include:

- Keypoint detection using `cv2.SIFT_create()`
- Feature matching using `cv2.BFMatcher()`
- Homography estimation using `cv2.findHomography()`
- Image transformation using `cv2.warpPerspective()`
- Blending for seamless transitions

Each step is chosen based on efficiency and robustness. Alternative methods and their potential impact on performance and accuracy are also discussed.

(a) First image (b) Second image

Figure 1: Two images side by side

2 Methodology

2.1 Keypoint Detection

The SIFT (Scale-Invariant Feature Transform) algorithm detects keypoints and computes descriptors for both images using:

```
sift = cv2.SIFT_create()
kp1, des1 = sift.detectAndCompute(img1, None)
kp2, des2 = sift.detectAndCompute(img2, None)
```

SIFT is chosen for its scale and rotation invariance.

(a) First image (b) Second image

Figure 2: Two images side by side

2.2 Feature Matching

Feature descriptors are matched using the Brute-Force Matcher (BFMatcher):

```
bf = cv2.BFMatcher()
matches = bf.knnMatch(des1, des2, k=2)
good_matches = [m for m, n in matches if m.distance < 0.75 * n.distance]
```

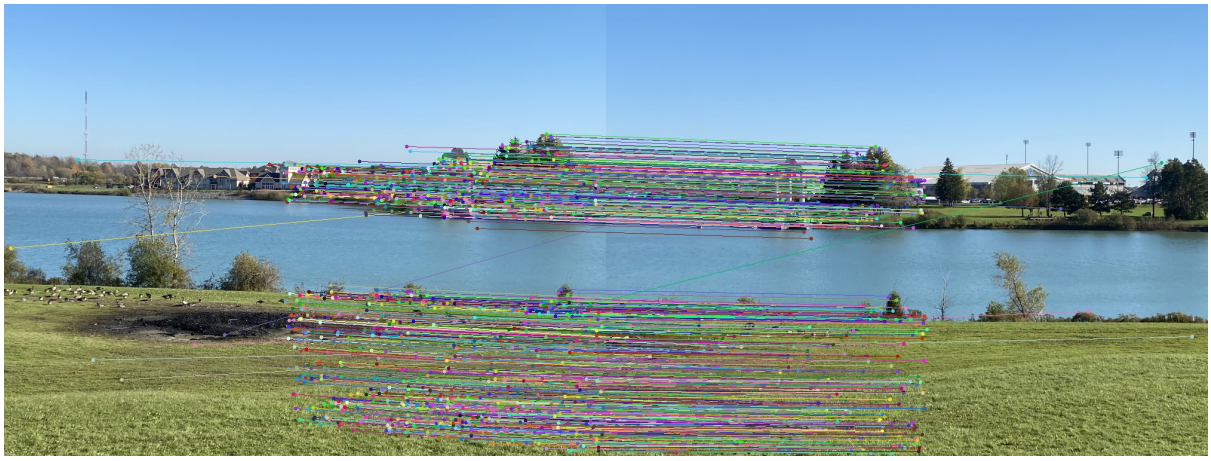

Figure 3: Your image caption

2.3 Homography Estimation using RANSAC

A transformation matrix (homography) is estimated using RANSAC:

```
pts1 = np.float32([kp1[m.queryIdx].pt for m in good_matches]).reshape(-1, 1, 2)
pts2 = np.float32([kp2[m.trainIdx].pt for m in good_matches]).reshape(-1, 1, 2)
H, mask = cv2.findHomography(pts1, pts2, cv2.RANSAC)
```

Without RANSAC, outliers would cause misalignment.

2.4 Image Warping and Stitching

Using the homography matrix, the second image is warped:

```
stitched = cv2.warpPerspective(img2, H, (width, height))
stitched[0:h1, 0:w1] = img1
```

2.5 Blending

A smooth blending mask is applied using:

```
mask = np.zeros_like(stitched, dtype=np.uint8)
mask[:, :w1] = 255
blended = cv2.seamlessClone(img1, stitched, mask, (w1 // 2, h1 // 2), cv2.NORMAL_CLON
```

3 Results and Discussion

The implemented pipeline successfully stitches images into a coherent panorama. Challenges include:

- Handling large transformations – mitigated by increasing feature matches.
- Managing varying lighting conditions – future work could explore histogram matching.

Figure 4: Your image caption

- Ensuring smooth blending – multi-band blending could enhance results.

Future improvements include exposure compensation, faster feature matching, and deep-learning-based feature extraction.

4 Conclusion

This report presents an image stitching pipeline using OpenCV functions such as `cv2.SIFT_create`, `cv2.BFMatcher`, `cv2.findHomography`, and `cv2.warpPerspective`. The choices were based on accuracy and robustness, and alternative methods were analyzed. The results demonstrate the effectiveness of this approach in generating seamless panoramic images.