Image Stitching

Ajitesh Kumar Singh

February 23, 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 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. It detects keypoints using a Difference of Gaussians (DoG) approach, which approximates the scale-normalized Laplacian of Gaussian:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma), \tag{1}$$

where $L(x, y, \sigma)$ is the image convolved with a Gaussian filter at scale σ. Local extrema in $D(x, y, \sigma)$ identify potential keypoints, which are refined by eliminating low-contrast and edge responses.

Each keypoint is assigned an orientation based on local gradients, and a 128-dimensional descriptor is computed using histograms of gradient orientations. This ensures robustness to illumination changes and minor affine transformations.

(a) First image (b) Second image

Figure 2: Two images side by side

2.2 Feature Matching

After detecting keypoints and computing descriptors, different feature matching techniques were explored:

- **FLANN (Fast Library for Approximate Nearest Neighbors)**: FLANN is an optimized approach for large datasets using tree-based indexing structures. However, it often resulted in incorrect matches due to its reliance on approximate nearest neighbors, making it less suitable for precise feature matching in this case.
- **KNN Matching with Lowe's Ratio Test**: KNN finds the two closest descriptors for each keypoint, and Lowe's ratio test discards ambiguous matches. However, in our case, it still led to mismatches due to noise and similar patterns in the images.

Why BFMatcher? Brute-Force Matcher (BFMatcher) was ultimately chosen due to its simplicity and accuracy. It computes the exact nearest neighbor for each descriptor, ensuring reliable matches. Given descriptors d_1, d_2 from two images, BFMatcher finds matches by minimizing the Euclidean distance:

$$\|d_1 - d_2\|_2 \tag{2}$$

Despite being computationally expensive, BFMatcher provided better results in this case, making it a suitable choice for precise feature correspondence.

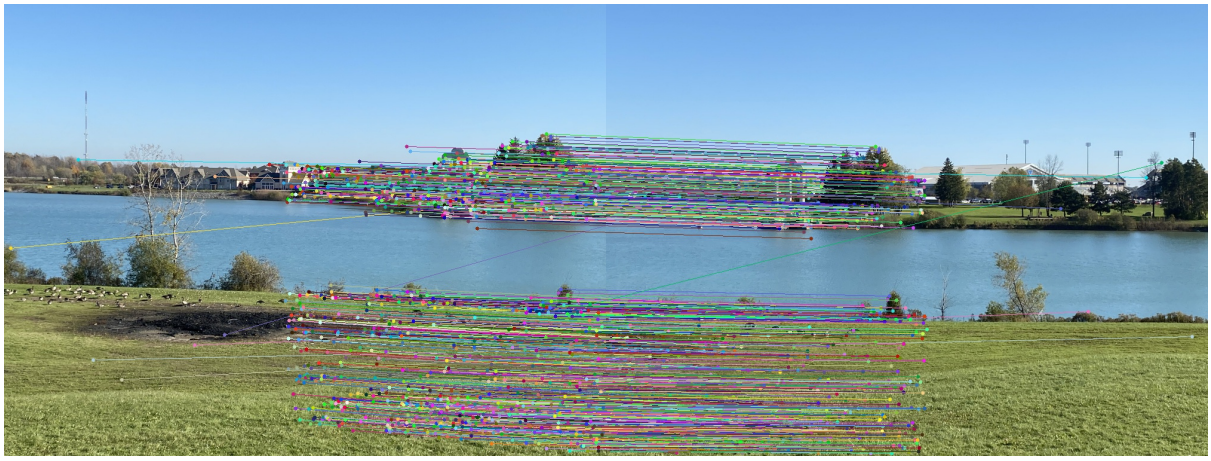

Figure 3: Your image caption

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

Image stitching involves aligning and blending multiple images into a single panorama. This is achieved by computing a homography matrix, which maps corresponding points between images using a perspective transformation.

2.4.1 Homography and Warping

Given a set of corresponding keypoints between two images, the homography matrix H is estimated as:

$$\mathbf{x}' = H\mathbf{x}, \tag{3}$$

where $\mathbf{x} = [x, y, 1]^T$ and $\mathbf{x}' = [x', y', 1]^T$ are homogeneous coordinates of corresponding points in the two images. The homography matrix H is a 3×3 matrix that encodes translation, rotation, scaling, and perspective distortion. It is computed using RANSAC to minimize the impact of outliers.

2.4.2 Warping the Second Image

Using the estimated homography, the second image is transformed to align with the first:

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

Here, `cv2.warpPerspective` applies the homography H to warp `img2` into the reference frame of `img1`, with the output image having dimensions `(width, height)`. The first image is then directly copied into the stitched result to preserve its original details.

2.4.3 Blending and Final Stitching

Since direct stitching may introduce visible seams, blending techniques such as multi-band blending or feathering can be applied to create a seamless transition between the images. This ensures that the final stitched image maintains smooth intensity variations and geometric consistency.

Figure 4: Your image caption

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.
- 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.