Computer Vision Assignment1

Aditya Shourya (i6353515)

May 13, 2024

Task 0.1: Assignment Tasks

This assignment is about developing your own panorama application by stitching together pairs of images, using concepts such as Harris Corner Detection, SIFT and RANSAC.

- Implemented our own Harris corner methods (with np,skimage and scipy.signals) and validated with opencv's implementation
- Successfully stitches 2 vertically split images in random order or slightly rotated and moved (affine transformed)
- Implemented our own ransac homography calculation (not validated with opencv)
- Works for > 2 inputs if iteratively passed.

Could not accomodate

- (Not asked) Does not work on horizontally split images (no vertical panorama)
- No calculation of Base Image for input size > 2.

Code at: `https://github.com/adishourya/image_stitching/tree/main`

Code Navigation

1. Image compilation for our experiments , hand calculating and validating Harris corners with openCV at `https://github.com/adishourya/image_stitching/blob/main/image_augmentation.ipynb`
2. library code for above in utils : `https://github.com/adishourya/image_stitching/tree/main/utils`
3. Ransac implementation and image stitching at : `https://github.com/adishourya/image_stitching/blob/main/stitching.ipynb`
4. For more detail check out the README file at : `https://github.com/adishourya/image_stitching/blob/main/README.md`

1 Harris Corners and SIFT

More Details about the transformations used and calculations used are in utils README : `https://github.com/adishourya/image_stitching/blob/main/utils/README.md`

1.0.1 Steps to calculate Harris corners

- convert image to greyscaled image
- Use Sobel Filters to calculate first order gradients
- Fomulate harris response by calculating square of the gradients (not second order) and $IxIy$
- Optionally apply non Maximum supression (Different than Maximum pooling used in convolutional network . Non Maximum Pooling is not a downsampling operation)

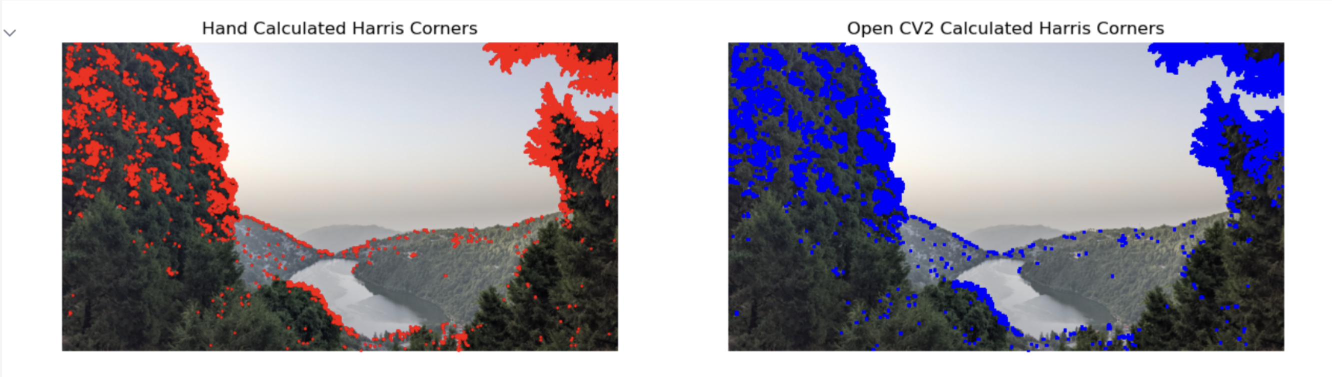

Figure 1: Validating Harris Corners with OpenCV

1.0.2 Parameters to tune in Harris Corners

These hyperparmaters can increase or decrease the number of harris corners generated .

- **Covariance Matrix (σ) in Gaussian Filter**
  - **Effect of Increasing Sigma:** Increasing σ in the Gaussian filter used in the calculation of I_{xx}, I_{yy}, and I_{xy} results in more smoothing. This can help suppress noise in the image and causes a loss of detail in the corner response,only capturing broad edges.which is ideal in performing image stitching. so we can get away with slightly high covariance matrix in this case
- **Thresholding**
  - **Effect of Increasing Threshold:** Similar to increasing the determinant of covariance matrix increasing the thresholding reduces number of harris corners. One might like to keep σ nominal but increase the threshold to get more robust harris corners.
- **Parameter k in Harris Response**
  - **Effect of Varying k:** The parameter k influences the sensitivity of the detector to edges versus corners. A small value of k (e.g., 0.04) typically provides good results, but varying it can shift the balance in detection, with lower values favoring more corner-like features and higher values favoring flatter regions.
  - **Challenges:** Selecting an appropriate k is essential for ensuring that the detector balances corner detection against the risk of responding too strongly to edges.
- **Application of Additional Filters on Gradient Computations**
  - **Effect of Different Filters:** Using different filters for the gradient computation, such as averaging filter instead of sobel, can affect the sensitivity and characteristics of the detected edges. This change depends on the filterbank used. performing one low pass filter on top of the highpass filters ; smoothens out the edges and removes finer details.
- **Non-Maximum Suppression Window Size**
  - **Effect of Increasing Size:** Increasing the window size for non-maximum suppression can help ensure that only the strongest corner within a larger neighborhood is detected, which can be useful in highly textured areas.
  - **Challenges:** Larger windows might suppress valid corners that are close to each other, thus reducing the detector's ability to resolve close features.

However, since SIFT requires scale and orientation information, which Harris corners, which detect major changes in picture intensity, do not naturally offer, we do not end up forwarding Harris corners to SIFT for descriptor extraction in our method. Scale-space extrema detection is used to identify features for which SIFT descriptors are improved, providing robustness against changes in scale and rotation. Using Harris corners directly in SIFT could lead to less stable or useful descriptors since they are usually localized without consideration for scale. Additionally, unlike the widely dispersed keypoints that SIFT normally handles, Harris corners may be unevenly distributed or densely concentrated throughout the image. In order to get around these problems, one may need to modify Harris corners by designating scale and orientation or use hybrid techniques that incorporate several characteristics

2 Steps taken in our RANSAC Process

Feature Detection and Matching Features in the images are detected using the Scale-Invariant Feature Transform (SIFT). Matches between these features are found using a FLANN-based matcher with a ratio test to filter out less reliable matches.

$$\text{good\_matches} = \left\{ m \mid \frac{m.distance}{n.distance} < 0.75 \right\} \tag{1}$$

Homography Estimation Using RANSAC RANSAC iteratively selects a random subset of matches, computes a homography, and then determines the number of inliers based on a predefined error threshold.See Appendix for calculation details A

Selecting Random Matches In each iteration, 4 matches are randomly selected to compute the homography.

$$\text{indices} = \text{random\_choice}(\text{len}(\text{points}), 4, \text{replace=False}) \tag{2}$$

Homography Computation Using these points, the homography matrix H is estimated.

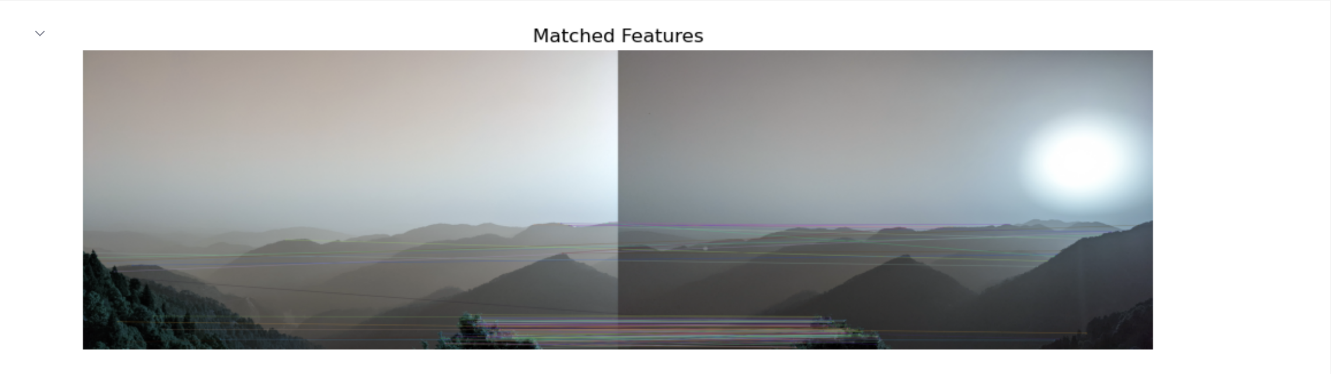

Figure 2: Matches pair

$$H, \_ = \text{cv.findHomography}(src\_pts, dst\_pts, 0) \tag{3}$$

Inlier Calculation The transformation H is used to project source points to the destination image, and the distance to the actual destination points is computed.

$$\text{projected\_pts} = \text{cv.perspectiveTransform}(src\_pts.reshape(-1, 1, 2), H) \tag{4}$$

$$\text{errors} = \|\text{projected\_pts} - dst\_pts\| \tag{5}$$

$$\text{inliers} = \text{errors} < \text{threshold} \tag{6}$$

Model Update If the current set has more inliers than any previous set, the model is updated.

$$\text{max\_inliers\_count} \leftarrow \text{current inlier count} \tag{7}$$

$$\text{best\_H} \leftarrow H \tag{8}$$

Final Model Refinement Using all the inliers from the best model, the final homography is re-estimated to refine the transformation.

$$\text{best\_H}, \_ = \text{cv.findHomography}(inlier\_src\_pts, inlier\_dst\_pts, \text{cv.RANSAC}) \tag{9}$$

Image Stitching Using the refined homography H, the images are warped and stitched together to form a continuous panorama.

$$\text{stitched\_image} = \text{cv.warpPerspective}(base\_image, best\_H, dimensions) \tag{10}$$

Post-processing Any black borders resulting from incorrect stitching are cropped to improve the visual quality of the panorama.

$$\text{cropped\_image} = \text{crop\_image\_only\_outside}(stitched\_image) \tag{11}$$

Appendices

A (Experiments) Cpython Approach : Hand calculating Homography

We used Random Sample Consensus (RANSAC) algorithm to estimate the homography between two images by finding the best transformation that aligns matched feature points. This approach is robust to outliers, which are common in feature matching.

A.1 Steps in RANSAC for Homography

A.1.1 Feature Detection and Matching

First, features are detected in both images using the Scale-Invariant Feature Transform (SIFT) and matched using the FLANN-based matcher. Good matches are filtered using the ratio test.

$$\text{good\_matches} = \left\{ m \mid \frac{m.distance}{n.distance} < 0.75 \right\} \tag{12}$$

A.1.2 Normalization of Points

To improve the accuracy and numerical stability, points are normalized such that the centroid is at the origin and the average distance to the origin is $\sqrt{2}$.

$$T = \begin{bmatrix} s & 0 & -s \cdot mean(x) \\ 0 & s & -s \cdot mean(y) \\ 0 & 0 & 1 \end{bmatrix} \tag{13}$$

where $s = \frac{\sqrt{2}}{std\_dev}$.

A.1.3 RANSAC Iterations

For each iteration, a subset of four matches is randomly selected to estimate the homography.

$$\text{Sampled Points: } src\_pts, dst\_pts \subseteq \text{Matches} \tag{14}$$

A.1.4 Homography Estimation with DLT

Using the Direct Linear Transformation (DLT) method, the homography matrix is estimated from the selected points.

$$A = \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & x'x & x'y & x' \\ 0 & 0 & 0 & -x & -y & -1 & y'x & y'y & y' \end{bmatrix} \tag{15}$$

$$\text{SVD}(A) \rightarrow \text{Homography Matrix } H \tag{16}$$

A.1.5 Projection and Error Calculation

Projected points are calculated and the distance to the corresponding destination points is used to count inliers.

$$\text{projected\_pts} = H \times src\_pts \tag{17}$$

$$\text{errors} = \|projected\_pts - dst\_pts\| \tag{18}$$

A.1.6 Model Update

If the current set has more inliers than any previous set, the model is updated.

$$\text{max\_inliers\_count} \leftarrow \text{current inlier count} \tag{19}$$

$$\text{best\_H} \leftarrow H \tag{20}$$

A.1.7 Final Homography Calculation

After all iterations, the homography matrix with the maximum inliers is re-calculated using all inliers to refine the model.

$$\text{best\_H}, \_ = \text{cv.findHomography}(inlier\_src\_pts, inlier\_dst\_pts, \text{cv.RANSAC}) \tag{21}$$

A.1.8 Image Stitching

The estimated homography matrix is then used to warp and stitch the images together, creating a panorama.

$$\text{stitched\_image} = \text{cv.warpPerspective}(base\_image, best\_H, dimensions) \tag{22}$$

B (Experiments) : Demonstrating on a samlpe image pair

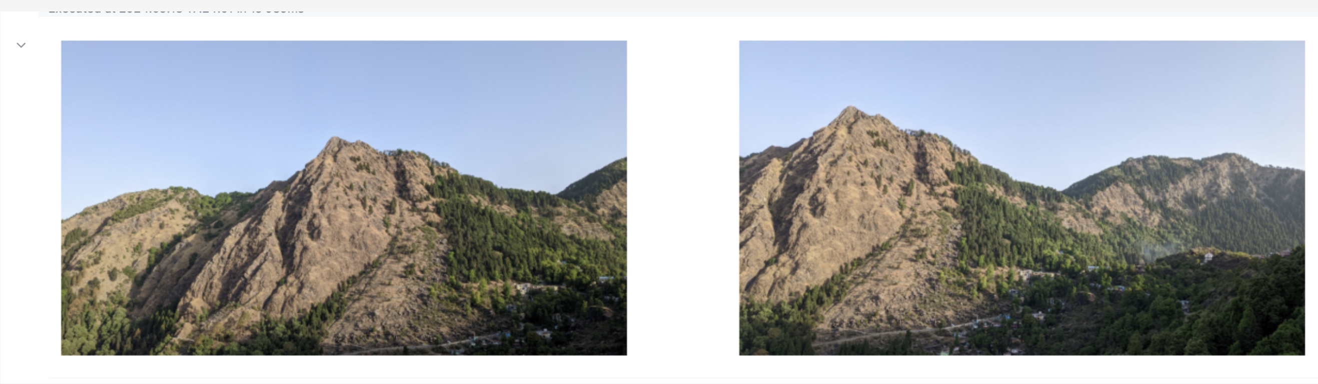

Figure 3: Input Pair

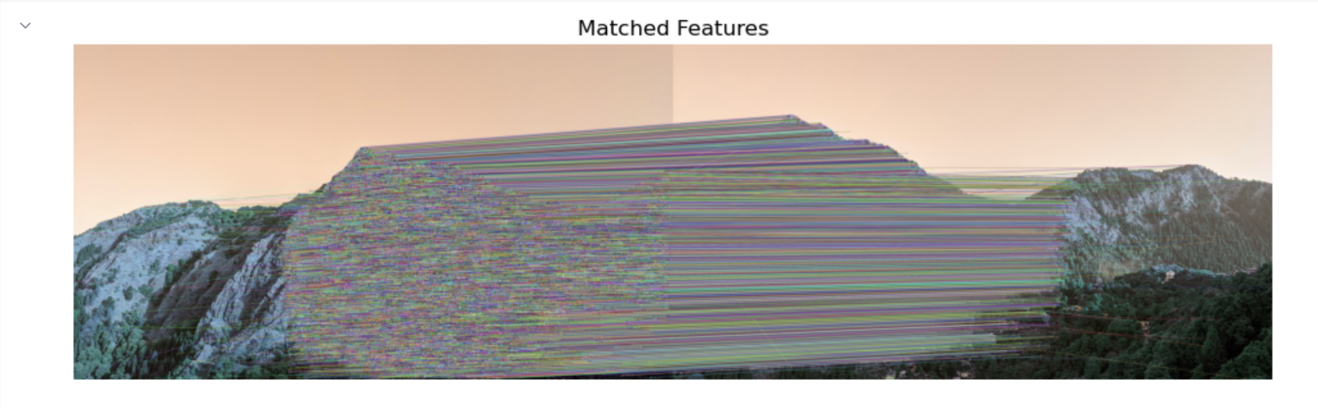

Figure 4: Calculated Mathces

Figure 5: Homography and warping