CS 456/556, Spring 2025 Programming Assignment 2 Augmented Reality with Planar Homographies

Srii Rohit Prakash - B01060706
CS 556, Spring 2025

March 18, 2025

1 Introduction and Overview

This project focuses on implementing an Augmented Reality (AR) system using planar homographies. The project is divided into the following parts:

- **Sparse Feature Matching**: Feature detection using the FAST detector, descriptor generation with BRIEF, and matching them using the Hamming distance.
- **Homography Estimation**: Using matched feature points, the project estimates the homography using RANSAC, computes essential matrices, and evaluates the results.
- **AR Application**: Using the computed homographies, we overlay virtual content onto real-world images or videos, creating an AR experience.

2 Sparse Feature Matching

2.1 FAST Detector - Q 3.1

The FAST (Features from Accelerated Segment Test) detector is used to detect feature points in images. Unlike the Harris corner detector, which uses eigenvalues of the structure

tensor, FAST uses a high-speed decision tree for quick corner detection. This makes FAST computationally more efficient in real-time applications. However, Harris provides better corner localization at the cost of speed.

Citations : Edward Rosten, Reid Porter, and Tom Drummond, "Faster and better: A machine learning approach to corner detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2010.

2.2 BRIEF Descriptor - Q 3.2

BRIEF (Binary Robust Independent Elementary Features) is a binary descriptor used to describe the image regions around detected feature points. The BRIEF descriptor compares pixel intensities in a local patch to form a binary string, which can then be used for fast matching using Hamming distance. Unlike filter-based methods, BRIEF evaluates pixel pairs directly, leading to a more efficient comparison.

Citations: Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua, "BRIEF: Binary robust independent elementary features," *European Conference on Computer Vision (ECCV)*, 2010.

2.3 Matching Methods - Q 3.3

Binary descriptors like BRIEF use the Hamming distance, which counts the number of differing bits between two binary strings, as the matching metric. This is much faster compared to Euclidean distance because it operates on binary values instead of floating-point values. Additionally, the BRIEF descriptors are memory-efficient, and nearest neighbor search using Hamming distance is optimized for bitwise operations.

Citations :

David G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision (IJCV)*, 2004. Herbert Bay, Andreas Ess, Tinne

Tuytelaars, and Luc Van Gool, "SURF: Speeded up robust features," *Computer Vision and Image Understanding (CVIU)*, 2008.

2.4 Feature Matching - Q 3.4

The `matchPics` function, implemented in `matchPics.py`, performs the following steps:

- Converts the input images to grayscale.
- Detects feature points using the FAST detector.
- Computes BRIEF descriptors for the detected keypoints.
- Matches the descriptors using Hamming distance.

We used `q3_4.py` to run the feature matching and visualize the results.

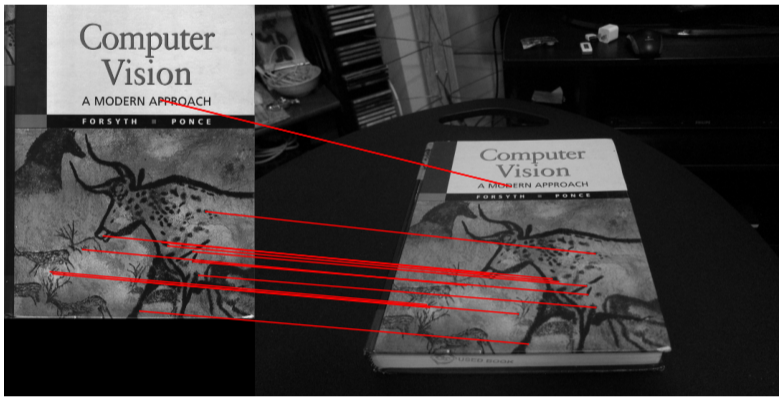


Figure 1: Feature matching results using BRIEF descriptors.

2.5 BRIEF and Rotations - Q 3.5

We tested the BRIEF descriptor on rotated versions of the image `cv_cover.jpg`, incrementing the rotation by 10 degrees. The number of feature matches was recorded and plotted in a histogram.

As expected, the number of matches significantly decreased with increasing rotation, indicating that BRIEF is not rotation invariant. This is due to BRIEF's reliance on intensity comparisons between pixel pairs, which are disrupted by rotation.

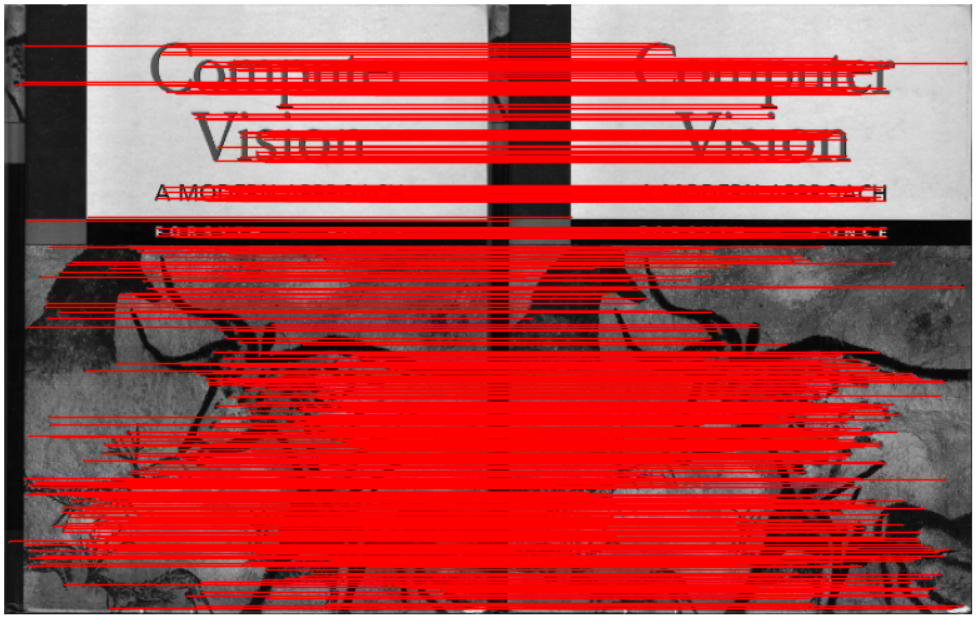
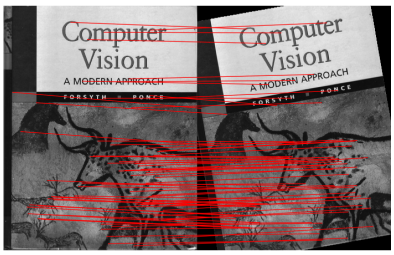
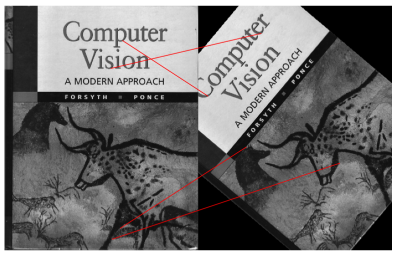


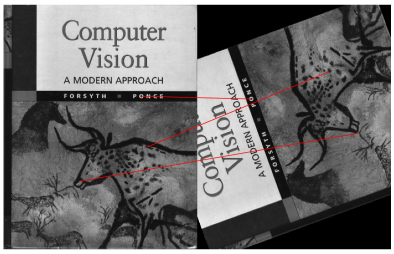
Figure 2: Feature matching results for rotated versions of the image.



(a) Angle 10 degree



(b) Angle 50 degree



(c) Angle 110 degree

Figure 3: 3D reconstruction from different angles.

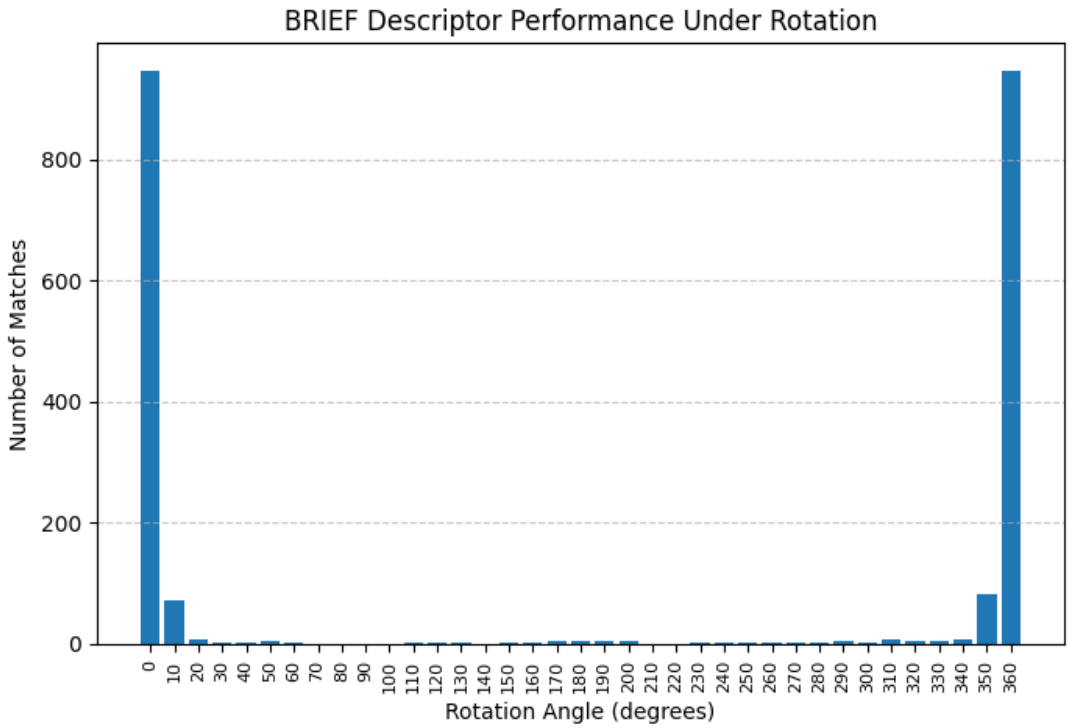


Figure 4: Histrogram.

3 Homography Estimation - Q 3.6

3.1 Fundamental and Essential Matrices

We first computed the fundamental matrix F using the eight-point algorithm, which maps points from one image to the epipolar lines in the other. The essential matrix E was then derived as $E = K_2^T F K_1$, where K_1 and K_2 are the intrinsic matrices of the two cameras.

$$E = \begin{bmatrix} -7.54935370 \times 10^{-3} & 9.88261784 \times 10^{-1} & 1.20159640 \times 10^{-1} \\ 5.11611775 \times 10^{-1} & -3.32218173 \times 10^{-3} & -5.74909705 \\ 1.80214556 \times 10^{-2} & 5.82251181 & 5.63155994 \times 10^{-3} \end{bmatrix}$$

3.2 Triangulation and 3D Reconstruction

Using the epipolar correspondences and camera poses, we triangulated the 2D points from both images to reconstruct 3D points. The quality of the reconstruction was evaluated using the re-projection error, which ideally should be below 2 pixels.

4 RANSAC for Homography Estimation - Q 3.7

RANSAC (Random Sample Consensus) is used to compute a robust homography by iterating through random subsets of the correspondences, computing candidate homographies, and determining the number of inliers for each. We implemented the function `computeH_ransac` in `planarH.py` to:

- Select four random correspondences.
- Compute a homography using `computeH_norm`.
- Evaluate the inliers based on a distance threshold.
- Iterate to find the homography with the highest inlier count.

5 HarryPotterize - Automated Homography Estimation - Q 3.8 , Q 3.9

The `HarryPotterize.py` script computes the homography between the book cover (`cv_cover.jpg`) and the AR source (`ar_source.mov`) by using `matchPics` and `computeH_ransac`. The warped image is then composited onto the desk image. However, due to aspect ratio differences between the book cover and the AR source, the warped image did not fully align with the book cover. To address this, we resized `hp_cover.jpg` to match the aspect ratio of the book cover before applying the homography.

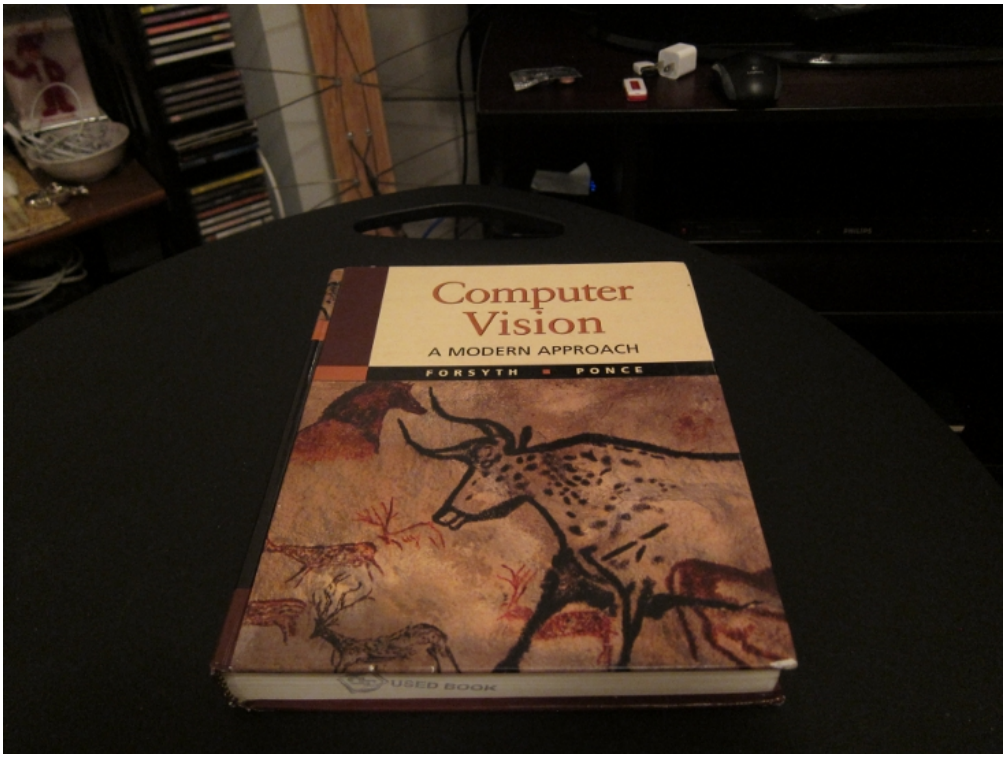


Figure 5: Text book.

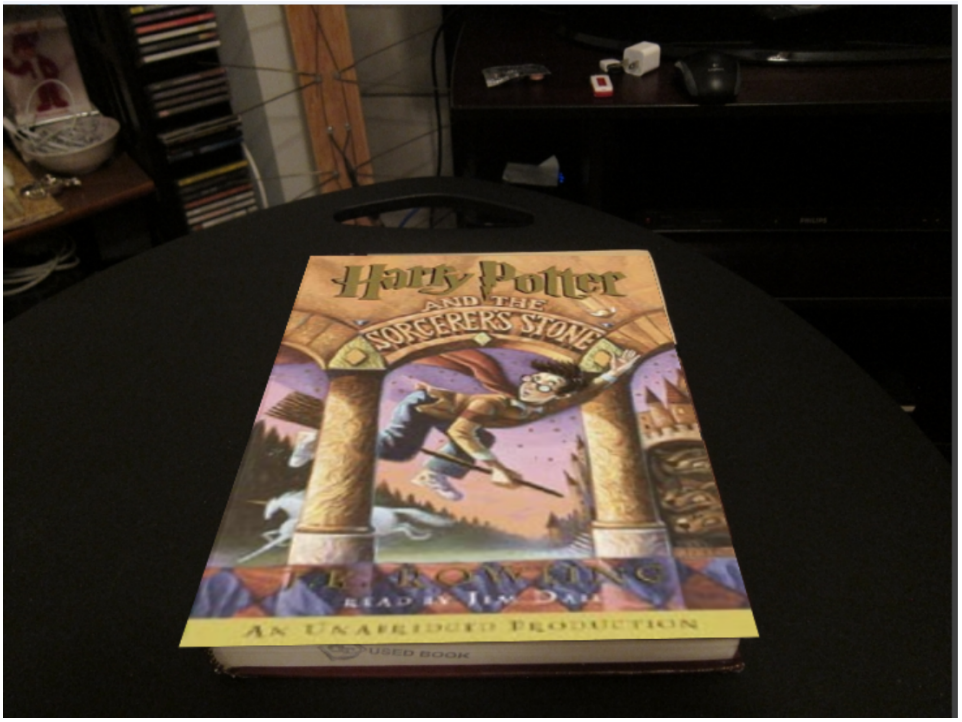


Figure 6: Result of HarryPotterize, compositing the warped image onto the desk.

6 Video Integration for AR - Q4.1

The `ar.py` script was written to track the book cover in each frame of `book.mov` and overlay the AR source from `ar_source.mov`. Cropping the frames was necessary to match the aspect ratio of the book cover, ensuring a realistic overlay.

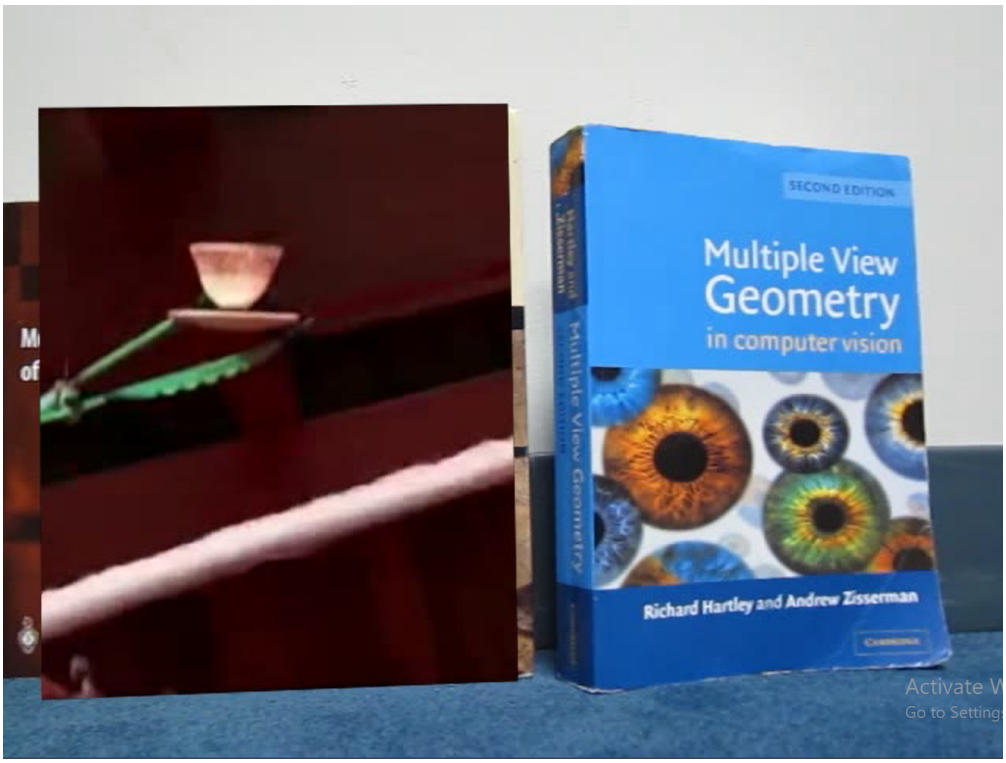


Figure 7: Final result of AR application, overlaying virtual content onto the real world.

7 Extra Credit: Panorama Creation - Q 5.1

For extra credit, we attempted to create a panorama by stitching two images. This was done by selecting matching points between the images, computing the homography, and warping the second image to align with the first. The result was a seamless panorama.

Figure 8: Image left.

Figure 9: Image right.

Figure 10: Final panorama result.

8 Conclusion

This project demonstrated the application of fundamental computer vision techniques, including feature matching, homography estimation, and augmented reality. The system successfully estimated homographies, applied them to images and videos, and created a functional AR application. Some challenges included handling feature matching in low-texture regions and correcting aspect ratio issues for proper AR overlay alignment.