Computer Vision Project Report

SIFT Feature Matching, Homography Estimation, Augmented Reality, and Image Stitching

---

1. Introduction

This project demonstrates a complete pipeline for applying **Augmented Reality (AR)** and **Image Stitching** using core computer vision techniques. The pipeline relies on:

- SIFT (Scale-Invariant Feature Transform) for detecting and describing keypoints
- Brute-Force Matching with Lowe's Ratio Test for finding correspondences
- Homography estimation using the Direct Linear Transformation (DLT) algorithm and Singular Value Decomposition (SVD)
- Inverse warping and bilinear interpolation for mapping pixel intensities
- Perspective transformation and masking for AR overlays

Part 1: Augmented Reality on a Book Cover

---

2. Feature Detection using SIFT

➤ *Overview of SIFT*

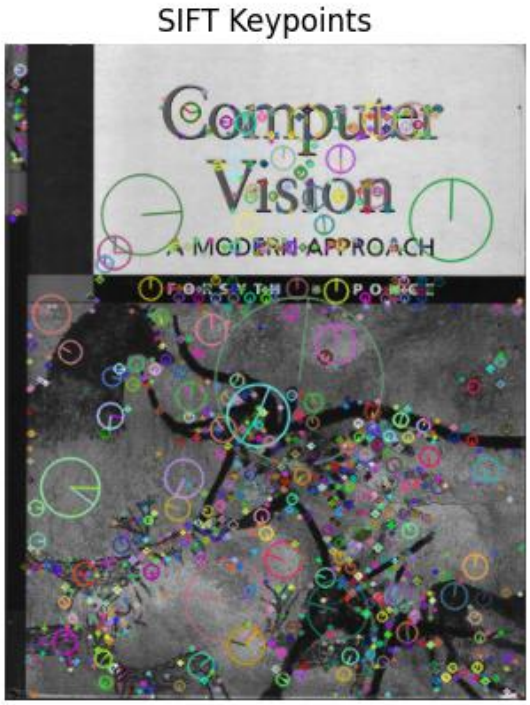
The SIFT algorithm detects local features in images that are invariant to scale, rotation, and illumination. It consists of several steps:

1. **Scale-space extrema detection**:
    - A Difference-of-Gaussians (DoG) is applied at multiple scales.
    - Keypoints are found as local extrema in the DoG pyramid.
2. **Keypoint localization**:
    - Fit a 3D quadratic function to localize keypoints with subpixel accuracy.
3. **Orientation assignment**:
    - Each keypoint is assigned an orientation based on the gradient direction histogram in its neighborhood.

4. **Descriptor generation**:
    - A 128-dimensional descriptor is built from gradient magnitudes and orientations around the keypoint.

➤ *Why SIFT?*

- It works on grayscale images because it depends on intensity gradients.
- It detects **edges, corners, and blobs**, making it suitable for tasks involving object recognition and matching.



---

3. Feature Matching using Brute Force and Lowe's Ratio Test

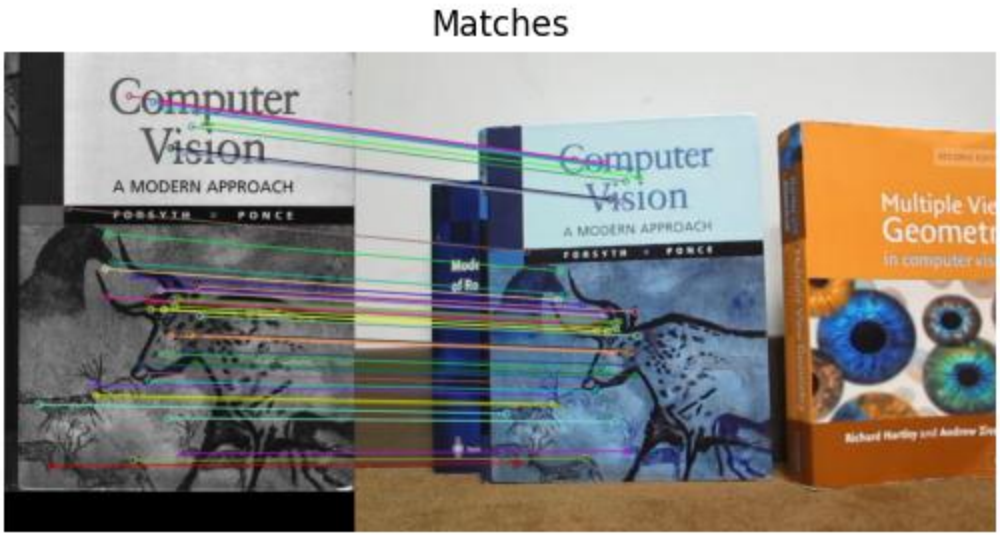
The Brute Force Matcher compares every descriptor in the first image with every descriptor in the second image using L2 distance. We use **k-Nearest Neighbors (k=2)** to find the two closest matches for each keypoint.

To filter out ambiguous matches, we apply **Lowe's Ratio Test**:

- For descriptors d_1 and its two closest matches m and n, retain the match only if:

  $\text{distance}(m) < \alpha \cdot \text{distance}(n)$

- In this implementation, $\alpha = 0.3$



---

4. Homography Estimation and Corner Detection

➤ *What is Homography?*

Homography H is a 3×3 transformation matrix that maps points from one image plane to another. It assumes that both images are related by a **projective transformation**, which occurs in planar scenes or under camera rotation.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

➤ *How is it Computed?*

We solve for H using the **Direct Linear Transformation (DLT)** algorithm. Given at least 4 point correspondences between two images:

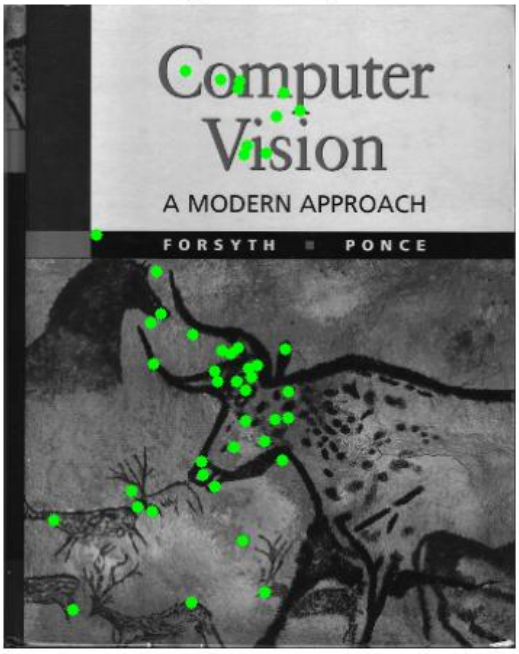
1. Construct a system of equations $A \cdot h = 0$ using the known source and destination points.
2. Solve using **Singular Value Decomposition (SVD)**.
3. Normalize the solution by setting $h_9 = 1$, then reshape into a 3×3 matrix.

➤ *Validation*

We compute the **reprojection error** to assess the accuracy of the homography:

$$\text{error} = \| H \cdot p - p' \|$$

Source Image with Original Points



Destination Image with Mapped Points

Mean reprojection error: 0.29 pixels

Corners on frame

5. Augmented Reality Overlay

Once H is known:

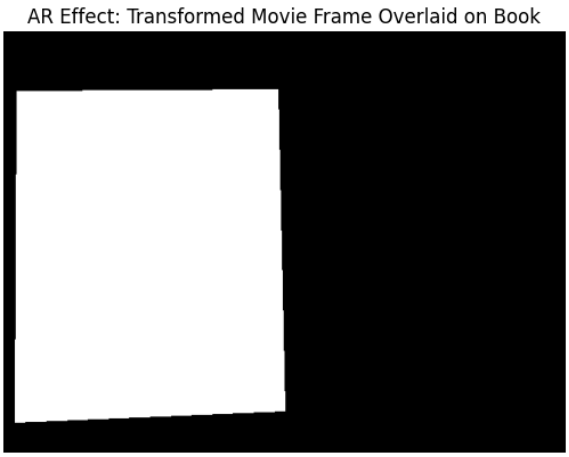
- Extract the corners of the book using perspective transform.
- Warp frames from ar_source.mov using cv.getPerspectiveTransform().
- Blend the warped video content over the detected book region using:
    - **Polygon masking**
    - **Bitwise operations** (for foreground/background separation)

---

6. Full AR Video Generation

The process is repeated for all frames of book.mov:

- Track the book location in each frame.
- Warp and overlay the corresponding frame from the AR source.
- Save the result as individual frames and compile them into a video.

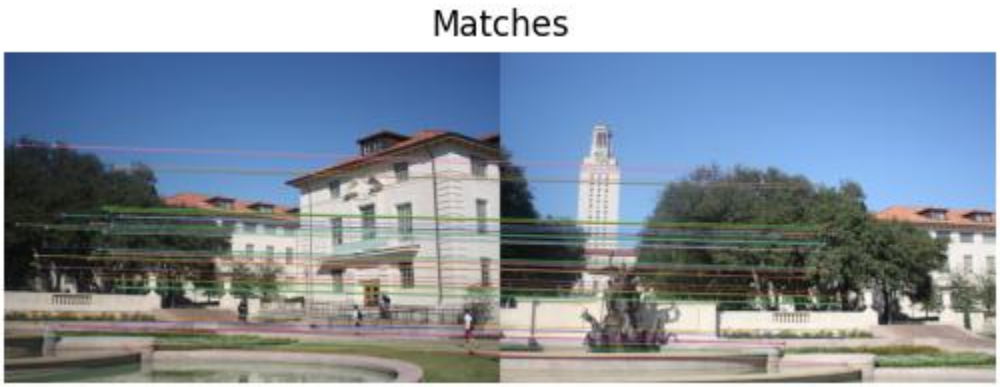


Part 2: Image Stitching and Panorama Creation

---

7. Image Matching and Homography

We apply the same SIFT + matching + homography pipeline to two partially overlapping panorama images:
pano_image1.jpg and pano_image2.jpg.



---

8. Inverse Warping with Bilinear Interpolation

➤ *Why Inverse Warping?*

Forward warping leads to **holes** (i.e., unassigned pixels) due to rounding errors and 1-to-many mappings. In contrast, **inverse warping** computes, for each output pixel, the corresponding source pixel via:

$$p = H^{-1} \cdot p'$$

We then fetch the pixel value from the source image using **bilinear interpolation** for subpixel accuracy.

➤ *Bilinear Interpolation Formula*

For a floating point source coordinate (x, y), the pixel value V is:

$$V = (1 - w_x)(1 - w_y) \cdot V_{00} + w_x(1 - w_y) \cdot V_{10} + (1 - w_x)w_y \cdot V_{01} + w_x w_y \cdot V_{11}$$

where:

- $w_x = x - \lfloor x \rfloor$, $w_y = y - \lfloor y \rfloor$
- $V_{00}, V_{10}, V_{01}, V_{11}$ are values at the four surrounding pixels.

---

9. Panorama Mosaic Construction

To combine the warped image2 with image1, we:

- Calculate output canvas dimensions to hold both images.
- Place both into the canvas using an offset to avoid negative coordinates.
- Overlap resolution: Only paste non-black pixels from the warped image (or apply blending for smoother transition).

Final Mosaic

10. Built-in Stitching for Comparison

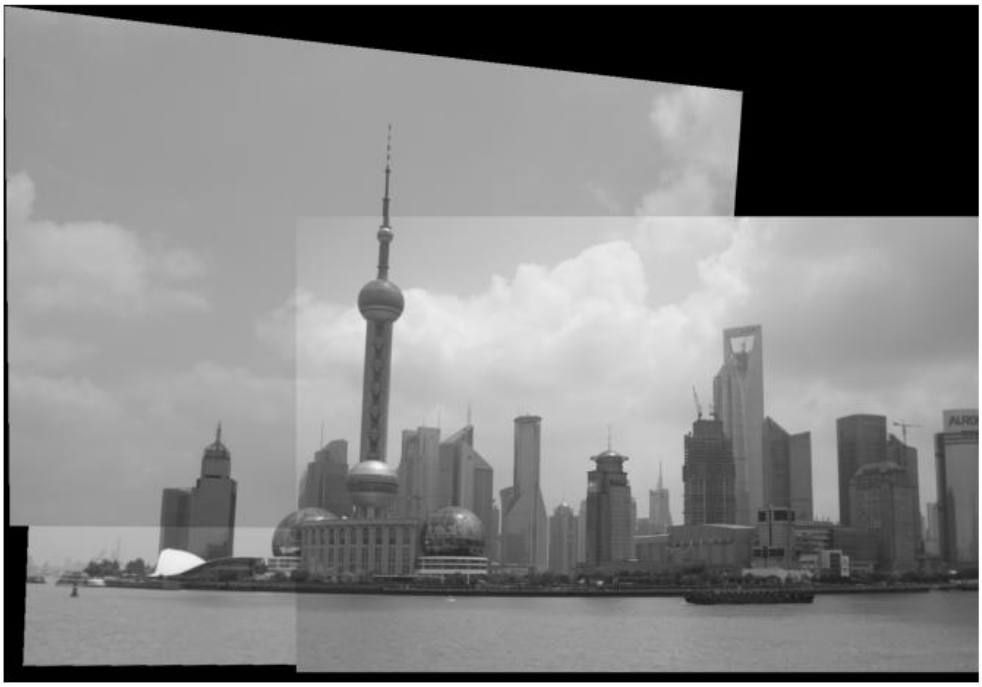
We use OpenCV's cv.Stitcher_create().stitch([image1, image2]) as a benchmark.

Final Mosaic

10. Bonus

Final Mosaic

Final Mosaic



11. Conclusion

This project illustrates the power of geometric transformations in computer vision:

- **SIFT** provides reliable features across scenes.
- **Homography** enables mapping between planar views.
- **AR overlays** demonstrate real-time applications.
- **Inverse warping** ensures accurate pixel mapping for stitching.
