



---

# COMPUTER VISION ASSIGNMENT 2

---



ABDELRHMAN TORK 8036  
AHMED HOSSAM NABIH 8116  
FELOUBATTEIR LEASHA 8231

# Part 1: Augmented Reality with Planar Homographies

## 1.1 Getting Correspondences

### Load and Preprocess Images

- The book image is loaded in grayscale using OpenCV to prepare it for feature detection.
- The first frame of the video is extracted using `cv2.VideoCapture`.
- After extracting the frame, it is converted to grayscale as well, since feature detection methods like SIFT work on intensity values rather than color information.

### Detect Keypoints and Compute Descriptors

- A SIFT (Scale-Invariant Feature Transform) detector is created using OpenCV's SIFT API.
- The `detectAndCompute` method is used to find keypoints and their corresponding descriptors in both the book image and the video frame.
- Keypoints represent distinctive locations in the image, and descriptors are numerical vectors that describe the region around each keypoint.

### Match Descriptors Using Brute-Force Matcher

- A Brute-Force matcher is initialized to match descriptors between the two images.
- KNN (K-Nearest Neighbors) matching is used with  $k=2$ , which returns the two best matches for each descriptor from the book image.
- This provides a list of potential matches with varying degrees of similarity.

### Apply Lowe's Ratio Test to Filter Matches

- Lowe's ratio test is applied to filter out weak or ambiguous matches.
- For each pair of matches  $(m, n)$ , the ratio of distances is calculated:  $m.distance / n.distance$ .
- If this ratio is less than 0.75, the match is considered good and added to the list of good matches.
- This step helps retain only the most confident and unique correspondences.

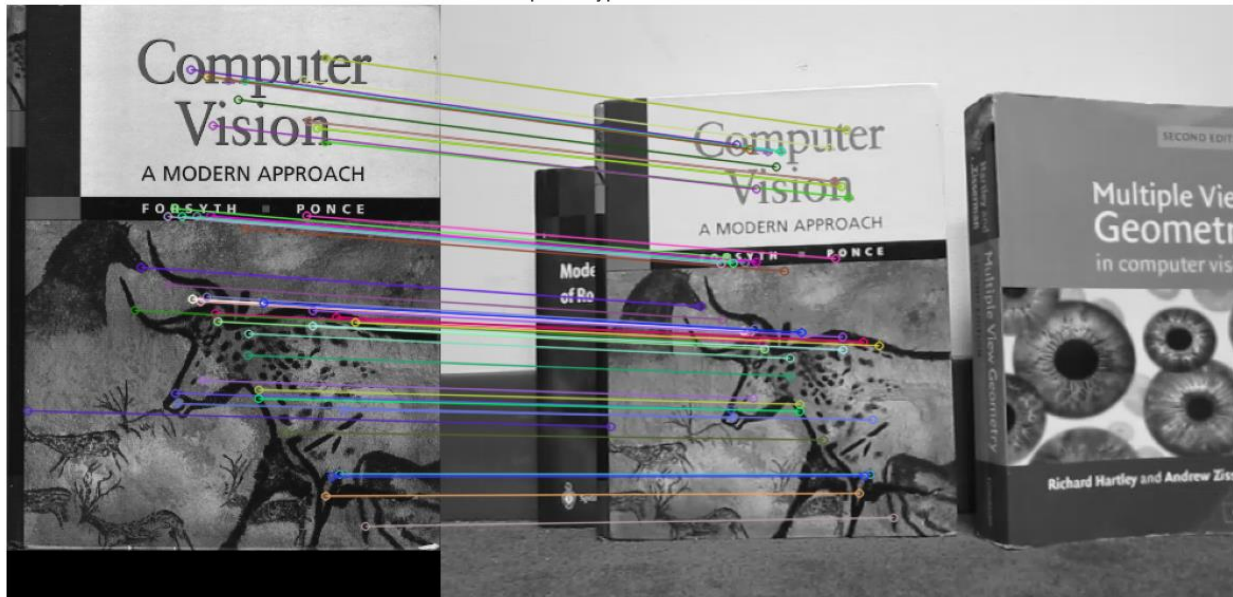
### Select Top 50 Good Matches

- The list of good matches is sorted based on the distance between descriptors, where a smaller distance means a better match.
- The top 50 matches with the smallest distances are selected for visualization.
- This ensures that only the most accurate and relevant correspondences are used.

## Draw and Display the Matches

- The selected matches are visualized by drawing lines between corresponding keypoints in the book image and the video frame.
- The `cv2.drawMatches` function is used to draw the top 50 matches.
- The result is displayed using a matplotlib plot, showing both images side by side with lines connecting matched keypoints.

Top 50 Keypoint Matches



## 1.2 Compute the Homography Parameters

### Extract Keypoints and Descriptors

- Convert both images to grayscale.
- Use SIFT to detect keypoints and compute descriptors.

### Match Descriptors

- Use Brute-Force Matcher with KNN ( $k=2$ ).
- Apply Lowe's ratio test to keep good matches.

### Estimate Homography (RANSAC)

- For a fixed number of iterations:
  - Randomly select 4 matching pairs.
  - Compute homography using least squares.
  - Apply homography to all points and calculate reprojection error.
  - Count inliers with error  $<$  threshold.
  - Keep the homography with the most inliers.

- Recompute final homography using all inliers.

### **Apply Homography**

- Convert 2D points to homogeneous coordinates.
- Multiply by homography matrix.
- Convert back to 2D.

### **Interactive Point Mapping**

- On mouse click in book image:
  - Transform point using  $H$  and draw on frame.
- On mouse click in frame:
  - Transform point using  $H^{-1}$  and draw on book.
- Display both images with drawn points.
- Exit on ESC key.

## **1.3 Calculate Book Coordinates**

### **Initialize Variables**

- Read the book image and video frames (book\_cap, ar\_cap).
- Extract keypoints and descriptors from the book image using SIFT.
- Define the corners of the book in the image (book\_corners).

### **Set Up Video Writer**

- Get frame properties (FPS, width, height) from the book video.
- Create a video writer for the output (output\_overlay.mp4).

### **Process Each Frame**

- Read frames from both the book and AR videos.
- Convert the book frame to grayscale and detect keypoints/descriptors.

### **Match Descriptors**

- Use KNN to match descriptors between the book and video frame.
- Apply Lowe's ratio test to filter good matches.

### **Compute Homography**

- If enough good matches exist, compute the homography matrix  $H$  using RANSAC.

### **Project Book Corners**

- Apply homography  $H$  to the corners of the book to project them onto the video frame.

### Overlay AR Frame onto Book

- Crop and resize the AR video frame to match the book image size.
- Warp the resized AR frame using the homography matrix.

### Mask and Overlay

- Create a mask for the book region in the video frame.
- Use the mask to overlay the warped AR frame onto the book region in the video frame.

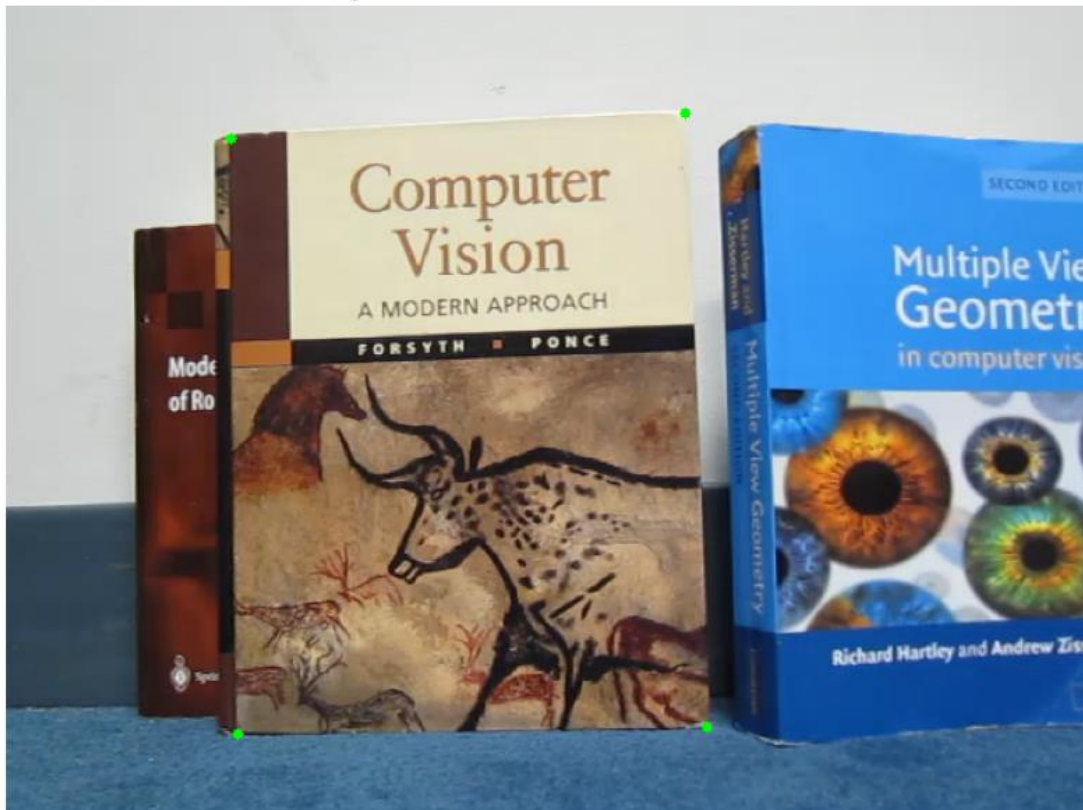
### Write Output

- Write the resulting frame (with overlay) to the output video.
- Repeat for all frames.

### Cleanup

- Release video captures and writer.

Projected Book Corners on Video Frame



## Part 2: Image Mosaics

### 2.1 Getting Correspondences and Compute the Homography Parameters

#### Load Images

- Read the two images (pano\_img1, pano\_img2) in grayscale for SIFT processing.

#### Match Keypoints

- Initialize SIFT detector.
- Detect keypoints and compute descriptors for both images.
- Use brute-force matcher to find the best matches between descriptors.
- Apply Lowe's ratio test to filter out good matches.
- Select top  $n$  good matches.

#### Draw Matches

- Use OpenCV to draw the matches between the images.
- Display the result using matplotlib.

#### Compute Homography

- Check if there are at least 4 good point correspondences.
- Set up a system of linear equations to solve for homography matrix  $H$ .
- Use least-squares to solve for the homography coefficients and reshape it into a  $3 \times 3$  matrix.

#### Apply Homography

- Convert source points to homogeneous coordinates.
- Apply the homography matrix to the points using matrix multiplication.
- Convert the transformed points back to Cartesian coordinates.

#### Draw Projected Points

- Project points onto the original image using the computed homography.
- Optionally, draw actual points (from the previous matching step) on the image for comparison.
- Display the image with the projected and actual points.

Top 50 Keypoint Matches



## 2.2 Warping Between Image Planes & 2.3 Create the output mosaic

### Warp Image Using Homography:

1. **Input:** Source image ( $src\_img$ ), homography matrix ( $H$ ).
2. **Step 1:** Calculate the corners of the source image.
3. **Step 2:** Apply the homography to the corners to get the warped corners.
4. **Step 3:** Compute the bounding box of the warped corners.
5. **Step 4:** Compute the output image dimensions.
6. **Step 5:** Build a destination grid (output image).
7. **Step 6:** Compute the inverse homography ( $H_{inv}$ ).
8. **Step 7:** Transform each point in the destination grid to the source image using inverse homography.
9. **Step 8:** Perform bilinear interpolation to sample pixel values from the source image and fill the destination image.
10. **Output:** Warped image.

### Create Image Mosaic:

1. **Input:** Two images ( $image1$ ,  $image2$ ), homography matrix ( $H$ ).
2. **Step 1:** Warp  $image1$  using  $H$ .
3. **Step 2:** Compute the mosaic size based on both images.
4. **Step 3:** Apply translation to center the warped image on the mosaic.
5. **Step 4:** Warp  $image1$  with the translated homography.
6. **Step 5:** Create a new canvas (mosaic).
7. **Step 6:** Overlay  $image1$  and  $image2$  onto the mosaic canvas.
8. **Output:** Final mosaic image

### Bilinear Interpolation (for sampling pixels):

1. **Input:** Image, x-coordinates, y-coordinates.
2. **Step 1:** Identify valid coordinates within image bounds.
3. **Step 2:** For each valid coordinate, calculate the four neighboring pixels and interpolate based on the distances.
4. **Output:** Interpolated pixel values for the destination image.

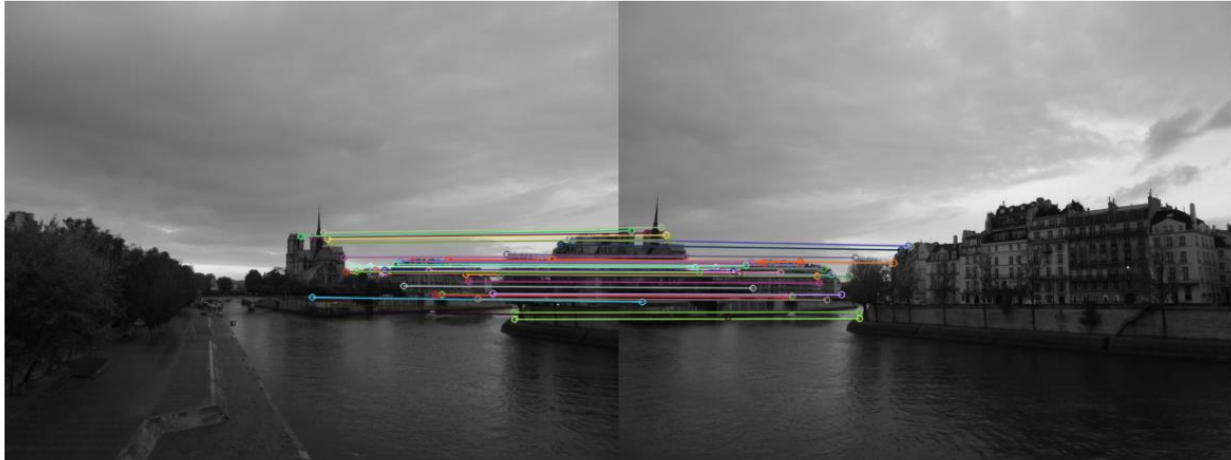
Final Mosaic





Another Image of Our choice (same steps)

Top 50 Keypoint Matches



Final Mosaic



## Part 3: Bonus

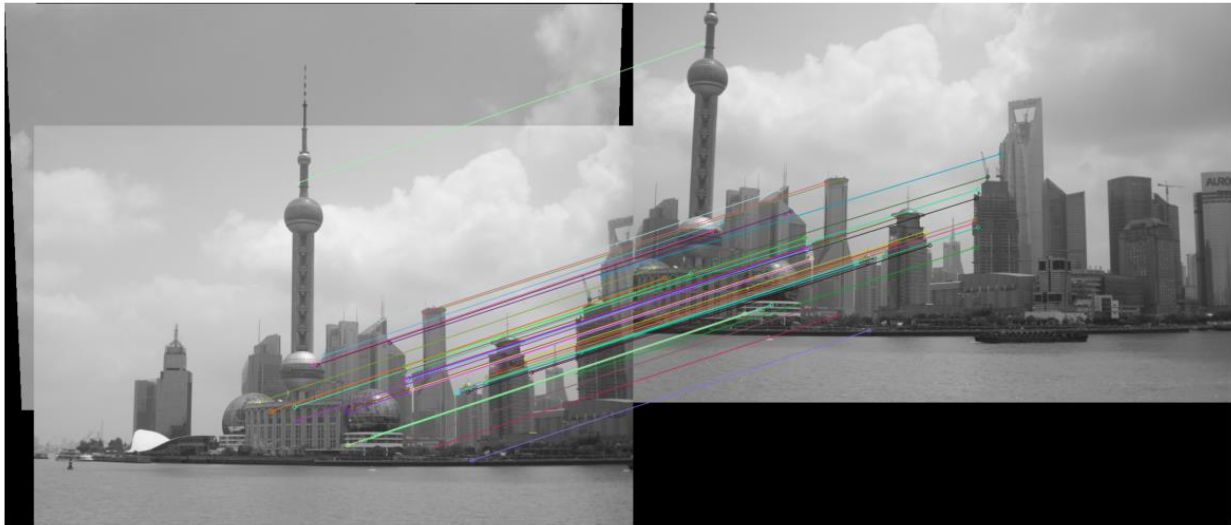
Top 50 Keypoint Matches



Final Mosaic



Top 50 Keypoint Matches



Final Mosaic

