BBM418-AIN433 : Computer Vision Laboratory
Fall 2025
Instructor: Assoc. Prof. Dr. Ali Seydi KEÇELİ
TA's: S. Meryem Taşyürek & M. Aslı Taşgetiren

Assignement 2: Homography Estimation

Due date: Friday, 14-11-2025, 11:59 PM.

The objective of this assignment is to develop a deep understanding of the geometric relationship between images through homography estimation. You will implement a complete pipeline for detecting keypoints, matching features, estimating the homography matrix using RANSAC, and warping one image into the coordinate frame of another. Finally, you will extend this process to create a panorama and augmented reality (AR) application.

This assignment emphasizes not only correct implementation but also conceptual understanding. You are expected to explain each step in your report, justify your design choices (e.g., matching strategy, RANSAC parameters), and interpret your results with clear reasoning and visuals.

1 Image Panorama Stitching

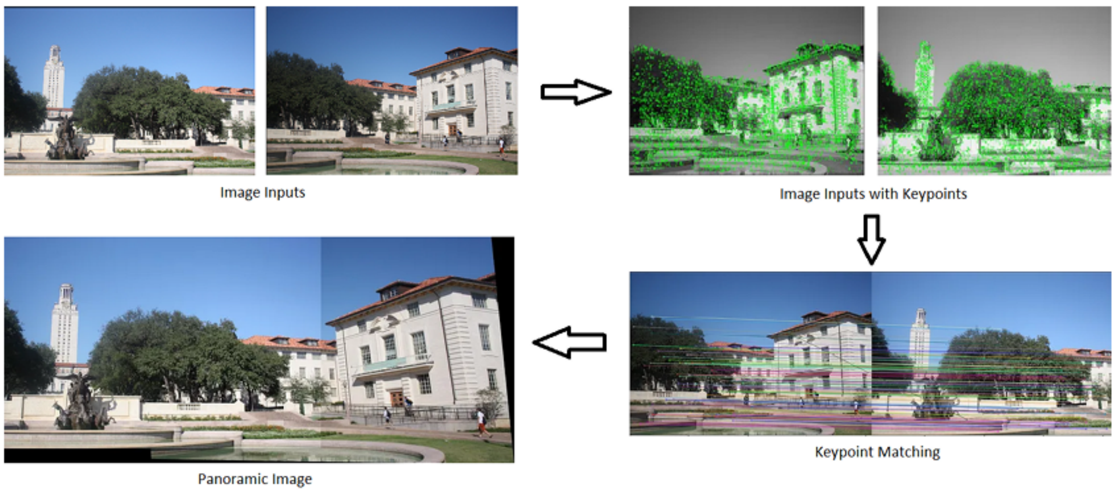

Figure 1: Merging images into panorama

Objective: In this part, you will work with **small image groups** that depict the same planar scene captured from slightly different viewpoints. Each scene contains a **reference image** and one or more **target images**, forming a dataset suitable for homography estimation and panorama construction.

Dataset: You will use a provided subset of the **HPatches** dataset of paired images located in the folder named `panorama_dataset/`. Each folder contains:

- two or more sub-images of the same scene (e.g., `v_bird/1.png`, `v_bird/2.png`, . . .),
- and a text file or matrix describing the **ground-truth homography** between the reference and target images (when available).

Your goal is to build a complete **image alignment pipeline** that estimates the homography between each image pair, uses it to warp the target image onto the reference image's coordinate plane, and finally merges them into a visually coherent panorama. The process involves the following main stages, each of which must be **implemented, visualized, and explained in your report**:

1. **Feature Extraction** – Detect interest points and compute local descriptors (SIFT, SURF, ORB).
2. **Feature Matching** – Match corresponding points between two images using distance-based similarity measures.
3. **Homography Estimation** – Compute the projective transformation matrix H from the matched correspondences.
4. **Image Warping and Stitching** – Apply the homography to transform one image into the other's plane and blend overlapping regions to form the panorama.

Implementation Details

You are expected to design and implement a complete pipeline for homography-based image alignment and panorama construction. Your implementation must be modular, well-documented, and visually demonstrative. Each major stage should be described, justified, and illustrated in your report with intermediate outputs.

Part 1: Feature Extraction (10 Points)

Detect interest points and compute local descriptors for each image using SIFT, SURF, and ORB methods. In your report discuss the properties of these detectors (e.g., scale and rotation invariance) and explain why such features are suitable for homography estimation. Visualize the detected keypoints on each image and comment on their spatial distribution.

Part 2: Feature Matching (10 Points)

Implement a matching procedure between image pairs using distance-based similarity (e.g., Euclidean or Hamming). You may use a k-nearest neighbor approach and apply Lowe's ratio test to filter ambiguous matches. In your report explain your choises. Visualize matched features by drawing correspondence lines between the two images. Explain how incorrect or uneven matching affects subsequent homography computation.

Part 3: Homography Estimation (20 Points)

In this part, you will implement the **Direct Linear Transform (DLT)** algorithm to compute the 3×3 projective transformation matrix H that maps points from one image to another. After estimating H from matched correspondences, apply the **RANSAC** algorithm to obtain a robust homography by rejecting outliers. You must implement this methods manually, use of library functions such as `cv2.findHomography()` is not allowed. Visualize inlier and outlier matches with different colors, and discuss how RANSAC improves the final estimation. You should:

- Derive and implement the DLT equations explicitly using your matched point pairs.
- Normalize point coordinates before estimation to improve numerical stability.
- Implement RANSAC to iteratively select minimal sets of points, estimate H, and retain the model with the highest inlier count.
- Report the number and percentage of inliers for each pair, and optionally compute the *reprojection error* as a quantitative measure of alignment accuracy.

Note: This implementation will be reused in later parts of the assignment, including the Augmented Reality task, where per-frame homographies must be estimated dynamically.

Part 4: Image Warping and Panorama Construction (15 Points)

Use the estimated homography to transform one image onto the coordinate system of the reference image. Apply geometric warping to align the images and create a seamless panorama. **There are 6 different scenes exist in the provided dataset. You should make panorama for each.** Discuss and experiment with different blending techniques such as simple averaging, feathering, or linear blending. In your report illustrate:

- the warped image before blending,
- overlapping regions between images,
- and the final panorama result.

2 Augmented Reality Application (25 Points)

In this final part, you will extend homography implementation to build a simple **Augmented Reality (AR)** application. Using your estimated homographies, you will project a video or dynamic image sequence onto a planar surface (such as a book cover) in another video, creating the visual effect of motion embedded in a real scene, ensuring that the overlaid content remains rigidly attached to the surface across frames.

Objective: Demonstrate the practical use of planar homographies in Augmented Reality by warping and compositing video frames onto a moving planar target. This part connects geometric vision concepts to a real-world application where each frame is aligned based on the computed homography. The resulting output should look as if a video is playing directly on the book's cover surface, maintaining the correct geometric perspective.

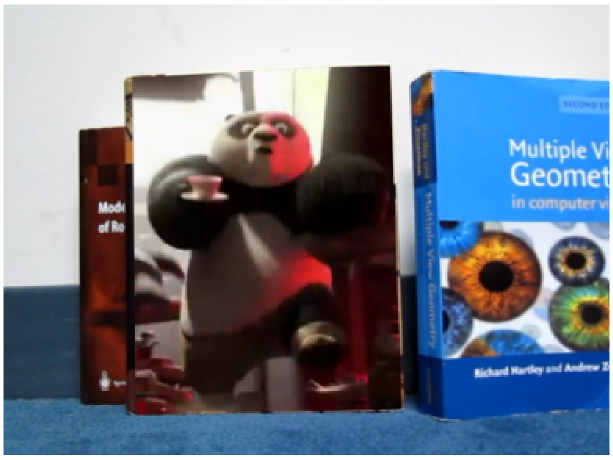

Figure 2: Rendering video on a moving target

Dataset: The dataset for this task is provided under the folder `ar_data/`. It contains the following files:

- `book.mov` – target video showing a moving planar surface (a book on a desk) captured from a changing viewpoint.
- `cv_cover.jpg` – reference image of the planar surface used for computing correspondences.
- `ar_source.mov` – a short video sequence to be projected onto the planar surface.

Implementation Steps:

1. Read both videos, `book.mov` and `ar_source.mov`, using `cv2.VideoCapture`.
2. For each frame t of the target video:
   - Detect and match feature points between `cv_cover.jpg` and the current frame of `book.mov`.
   - Estimate the homography H_t using your DLT + RANSAC implementation.
   - Read the corresponding frame from `ar_source.mov` (loop the video if necessary).
   - Warp the source frame using H_t to align it with the book surface.
   - Composite the warped frame onto the current target frame produce the AR result.
3. Save all augmented frames as a single output video (e.g., `ar_dynamic_result.mp4`) using `cv2.VideoWriter`.

Note: The book and the videos we have provided have very different aspect ratios (the ratio of the image width to the image height). You must crop each frame to fit onto the book cover. You must crop that image such that only the central region of the image is used in the final output. Also, two videos have different lengths, you can use the shorter length for your video.

Expected Outputs:

- A final AR video (`ar_dynamic_result.mp4`) where the projected video appears rigidly attached to the moving book surface.
- Representative frames in the report illustrating successful alignment under different perspectives.
- A concise explanation (5–6 sentences) describing your per-frame homography estimation and video compositing procedure.

Hints:

- Limit processing to a subset of frames (e.g., every 2nd or 3rd frame) for efficiency, if needed.
- Optionally, use feature tracking (`cv2.calcOpticalFlowPyrLK`) to propagate matches between consecutive frames and reduce flicker.
- Ensure consistent scaling between the source and target frames before warping.
- Verify that the overlaid content does not drift or misalign across frames.
- Note that this is a time-intensive job and may take many hours on a single core. Debug before running your full script (e.g. by saving a few early AR frames and verifying that they look correct).

Evaluation Criteria (20 Points):

- (7 pts) Correct per-frame homography estimation and temporal consistency.
- (7 pts) Accurate and stable projection of the source video onto the target surface.
- (5 pts) Smooth and visually coherent final AR video.
- (6 pts) Clear explanation and representative results (frames + discussion) in the report.

Example Application: You may project a moving video or animation onto a planar book cover or a computer monitor, creating an effect similar to "interactive posters" or "Harry Potter–style newspapers." This demonstrates how planar homographies enable dynamic overlays in real scenes, bridging the concepts of image alignment and Augmented Reality.

3 Report and Analysis (20 Points)

Submit a single, well-structured `report.pdf` that explains *what you built*, *why you made the choices you did*, and *how you verified the results*. Keep the focus on clear reasoning and visual evidence rather than heavy metrics.

Required Structure

1. **Overview (1–2 paragraphs).** Briefly state the goal of the assignment and summarize your full pipeline.

2. **Dataset & Setup.** Datasets used, folder layout, resolution/frame-rate choices (for videos), and any preprocessing.

3. **Methods (by Part).**
   - **Feature Extraction:** detector/descriptor choices (SIFT/SURF/ORB), parameter settings, and rationale.
   - **Feature Matching:** matcher type (e.g., k-NN), ratio test threshold, cross-check, and typical failure modes you observed.
   - **Homography Estimation:** brief DLT recap, RANSAC design (sample size, iterations, inlier threshold, scoring), and normalization if used.
   - **Warping & Panorama:** warping direction(s), canvas sizing, blending choice (copy/average/linear), and handling of overlaps.
   - **Augmented Reality (AR):** per-frame homography strategy (match vs. track), aspect-ratio handling/cropping for the source video, compositing approach, and any speed/stability tricks.

4. **Results (Figures & Tables).** Provide the following at minimum:
   - *Detected keypoints* (overlaid on images).
   - *Matched correspondences* before/after filtering (Lowe ratio/cross-check).
   - *Inlier vs. outlier visualization* under RANSAC (different colors).
   - *Warping & panorama:* warped image overlay (with transparency), overlap mask, and final panorama. You should show this for every scene in the dataset.
   - *AR:* at least three representative frames (beginning/middle/end) from your AR video showing the projected content geometrically aligned.
   - *Comparison table:* SIFT/SURF/ORB with runtime and inlier ratio (per scene). *Optional:* add reprojection error and/or GT comparison if available.
   - *Ablation snapshot (concise):* one small table or paragraph on how RANSAC parameters (iterations, inlier threshold) affected inlier ratio or visual alignment.

5. **Discussion (Accuracy, Robustness, Limitations).** When does the pipeline work well vs. fail? Comment on planarity assumptions, parallax, repetitive textures, illumination changes, motion blur, and frame-to-frame stability in AR.

6. **Reproducibility Notes.** List key parameters (detector thresholds, ratio test value, RANSAC iterations/threshold), random seeds, and any hardware/GPU notes. Include a short command-line example to reproduce your main results.

Evaluation Criteria (20 Points)

- (8 pts) Clarity and depth of explanations (methods & choices well-justified; concise math where appropriate).
- (8 pts) Quality and completeness of visuals (all required figures; legible, well-labeled, representative).
- (4 pts) Comparative insight and reflection (meaningful table(s); brief but thoughtful discussion of results/limitations).

What to Hand In

Your submission must include the following files:

- `report.pdf` – a well-structured written report following the required format in Section 5.
- `b<studentNumber>.ipynb` (or equivalent `.py` scripts) – your implementation code, organized and properly commented.
- `panorama_results` – your final panorama images in a folder.

Submission Format: Package all files into a single archive named:

`b<studentNumber>.zip`

and submit it to the course submission system:

`https://submit.cs.hacettepe.edu.tr`

Video Deliverables (Required via External Link) Due to file-size limitations on the submission system (maximum 8 MB), all video outputs must be uploaded to an external storage service such as Google Drive, OneDrive, or Dropbox. You must make the videos accessible via public or "anyone with the link" sharing settings and include the links in your `report.pdf`.

The following videos are required:

- `ar_dynamic_result.mp4` – your final Augmented Reality video demonstrating dynamic source + dynamic target projection.
- `demo.mp4` – your 5-minute presentation video summarizing your report, visuals, and implementation.

Clearly label each video link in your report, for example:

```
AR Video Result: https://drive.google.com/...
Demo Presentation: https://drive.google.com/...
```

Demo Video Requirements (Mandatory, 5 Minutes Maximum) Prepare a concise presentation video (`demo.mp4`) summarizing your work. It should primarily focus on your `report.pdf` visuals and overall pipeline, while briefly showing implementation details when relevant.
Your demo should:

- Present the overall goal of the assignment and key steps of your pipeline (feature extraction, matching, homography estimation, panorama, and AR).
- Show selected figures and visuals from your report to explain results and observations.
- Demonstrate your final outputs, including the dynamic AR video and panoramas.
- You may show short snippets of running code or intermediate outputs to clarify specific implementation aspects.
- Keep the presentation clear, self-contained, and under 5 minutes.

Tip: You may use tools such as Zoom recording, or PowerPoint screen recording to capture your screen and narration together. Ensure that audio is clear and visuals are readable at normal playback speed.

Checklist Before Submission:

- All figures, tables, and visuals in `report.pdf` are clearly labeled and referenced.
- Your code executes without errors and reproduces your results.
- `ar_dynamic_result.mp4` demonstrates a stable and realistic AR projection, and a working link to this video included in your `report.pdf`.
- `demo.mp4` presents your report and results clearly within 5 minutes, and a working link to this video included in your `report.pdf`.

No additional documents are required. Your written report and presentation video together must demonstrate both your conceptual understanding and practical implementation of the assignment.

Academic Integrity

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for online/AI sources. Make use of them responsibly, refrain from generating the code you are asked to implement. Remember that we also have have access to such tools, making it easier to detect such cases.