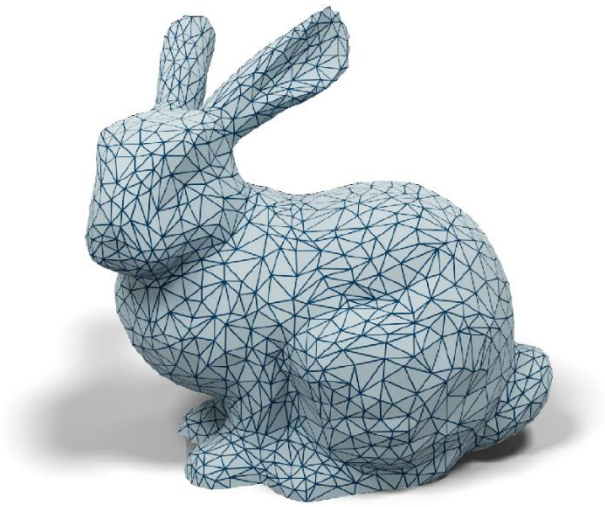

3D Computer Vision

Dr. Ahad Harati

Homework 2:

Plane Detection and Tracking via Homography Estimation

Designed By:

Mahdieh Alizadeh - Ali Goli

Mahdieh20201@gmail.com

Spring 2025

Preface

Welcome!

This assignment focuses on extracting and tracking planar surfaces in a real-world image sequence using classical computer vision methods. You will work with the "Office Maze" sequence from the TUM RGB-D dataset, which provides grayscale images of an indoor environment captured with a calibrated camera. Despite the absence of RGB or depth information, planar structures such as walls, floors, and desks can still be robustly identified by exploiting geometric constraints.

In the first part of the task, you are required to detect planes by computing homographies between pairs of consecutive frames. Homographies describe the transformation of points between two views of the same plane and can be estimated from matched feature points. By clustering feature correspondences that conform to a common homography, you can identify sets of features that likely belong to the same planar surface. Each plane should be visualized using a distinct color over the grayscale image, allowing for easy visual identification.

The second part of the task builds upon the first. Once planes are detected, they should be tracked across the sequence. You will design a tracking system that assigns consistent identities to planes across frames, ensuring that each plane retains its color and ID over time. This requires associating new observations with existing plane hypotheses, potentially using homography re-estimation, feature matching, or motion consistency. The ability to track planes robustly is a step toward building higher-level understanding of 3D scenes over time.

Note:

• Include well-commented code and relevant plots in your notebook.

• Clearly present all comparisons and analyses in your report.

• Ensure reproducibility by specifying all dependencies and configurations.

• Real-time processing bonus is offered for systems that operate at or above 15 frames per second.

DataSet

Office Maze from TUM is provided as your test dataset includes gray scale images and calibration parameters of the camera.

Submission

The deadline for this homework is 1404/02/31 (May 21th 2025) at 11:59 PM

Please submit your work by following the instructions below:

- Place your solution alongside the Jupyter notebook(s)
    - Your Written Solution must be a single PDF file named *HW2_Solution.pdf*

    - If there is more than one Jupyter notebook, put them in a folder named Notebooks
- Zip all files together with the following naming format: *C3V_HW2_[Fullname].zip*
- If you have any questions about this homework, please ask them in our Telegram Group.
- If you are using any references to write your answers, consulting anyone, or using AI, please mention them in the appropriate section.

Grading:

The grading will be based on the following criteria, with a total of 100 points:

Tasks	points
Task 1: Implemetation and Results	45
Task 2: Implementation and Results	45
Clarity and Quality of Code	5
Clarity and Quality of Report	5
Bonus: real-time algorithm (15-20 fps)	+10

Keep up the great work and best of luck with your submission! 😊

Task 1.

Plane Detection Using Homographies

Instructions:

- For each pair of consecutive frames, extract feature correspondences using any classical method: **SIFT**, **ORB**, or **GoodFeaturesToTrack** + **KLT**.
- Estimate one or more **homographies** that group feature points into **planar regions**.
- Use these homographies to segment the image **into distinct planar regions**.
- Assign a unique random color to each newly detected plane and visualize the plane's features on the grayscale image using that color.

Note:

- Do not use any deep learning methods.
- You can use OpenCV library.

Hints: To detect planar regions in the image, you may iteratively apply **homography** estimation using **RANSAC**. In each iteration, identify feature correspondences that are consistent with a single homography, indicating a potential plane. Once a set of inlier features corresponding to a plane is found, remove those features from the pool and repeat the process on the remaining features to detect additional planes.

Deliverables:

- Python implementation for plane detection using homographies.
- Visualization: grayscale images overlaid with color-coded feature points, where each plane has a distinct color.
- Short report describing your approach, algorithm choices, and output examples.

Task 2.

Plane Tracking Across Frames

Instructions:

- ❖ For each plane detected in Task 1, implement a plane tracker that follows the same plane across subsequent frames.
- ❖ Each plane should retain the same ID and visualization color across frames where it is visible.
- ❖ Plane tracking strategy is your choice, as long as it is classical (e.g., using feature descriptors or optical flow).

Note:

- Do not use deep learning or pretrained models.
- Your plane tracker must correctly match newly detected planes to previously seen ones (based on feature overlap, homography consistency, etc.) and assign the correct ID (i.e., keep the same color). If a new plane is detected, assign it a new ID and color.

Visualization Output:

- Grayscale images with overlaid, color-coded feature points belonging to tracked planes.
- As planes persist across frames, their visual color must remain consistent.
- New planes appearing in the sequence must be assigned new colors.

Deliverables:

- Python implementation.
- Short report describing your approach, algorithm choices, and output examples.

Bonus: Real-Time Performance (Optional):

- If your full system (**plane detection** + **tracking**) runs at 15–20 FPS or higher, you will receive a bonus.
- You may optimize performance by:

- Pre-loading or pre-processing future frames and features in a queue using a separate thread.
- Efficient feature tracking using KLT or optimized matchers.
- Avoiding overhead in visualization and I/O.