Assignment 1: Augmented Reality (80 marks + 10 marks bonus)

COL7680/JRL7680/COL780: Introduction to Computer Vision

Due on 15 Feb 2026 11:59pm

1 General Instructions

1.1 Academic Integrity

This assignment must be completed individually. Any form of plagiarism or academic dishonesty will be addressed in accordance with institute policy and may result in severe penalties.

1.2 Evaluation Process

All submissions will be subject to demonstration sessions (demos) where students will be required to explain their understanding of the underlying concepts and demonstrate their implementation. Students should be prepared to discuss their solution approach and answer questions about their code.

1.3 Collaboration and Discussion

While the assignment is individual work, students are encouraged to engage in healthy academic discussion regarding conceptual understanding, approach strategies, and technical challenges. Please utilize the course Piazza forum for:

- Discussing high-level ideas and concepts
- Seeking clarification on assignment requirements
- Troubleshooting technical issues and bottlenecks

Course Piazza: `https://piazza.com/iitd.ac.in/spring2026/col7680`

1.4 Submission Requirements

Your submission must include a comprehensive solution report that documents:

- Detailed explanation of your implementation approach
- Key design decisions and their justifications
- Challenges encountered and how they were addressed

- Discussion forums, documentation, and online resources consulted during the assignment
- Any assumptions made in your solution

Please ensure all cited resources are properly referenced in your report.

2 Objective

The goal of this assignment is to develop a pipeline to detect custom AR tags in video/webcam streams and use these detections to perform Augmented Reality(AR) tasks like 2D image overlay and 3D object projection. All required content for the assignment can be downloaded from here.

3 Dataset

You are provided with multiple AR tags drive link. Helper function to create new AR Tags is provided in the *utils* script. Please print them on a sheet of paper and perform the following tasks on your webcam stream and provided video(s). Your solution must be robust enough to handle varying camera angles and multiple tags within the same frame.

4 AR Tag Information Representation

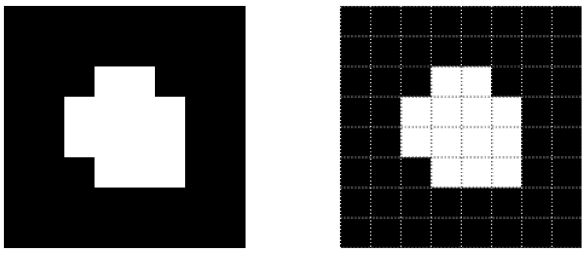

Figure 1: Custom AR Tag in a grid of 8x8

The custom AR tags function as fiducial markers by encoding spatial and numerical data within a structured 8×8 grid. There is a 2 cell width solid black outer border which provides high contrast for tag detection. The four outer corners define the tag's boundary and can be used to compute the homography required to warp the tag into orthographic view. The information about the tag is contained within the internal 4×4 grid, represented as follows:

- **Orientation Marker:** A white cell is placed at bottom right cell of the internal 4×4 grid. By locating this marker, the system determines the tag's upright orientation, ensuring that AR content is superimposed with the correct alignment.
- **Tag ID:** The core identifier is stored in a central 2×2 matrix. Each cell in the grid represents a binary bit, where white cells denote a value of 1 and black cells denote a value of 0. This 4-bit signature, in clock-wise pattern, allows for unique identification of different markers in the same frame.

5 Task 1: AR-Tag Detection and Identification

Implement a detection pipeline to extract AR Tag(s) from each frame of the stream. Once a tag is detected, you must extract its ID and determine its orientation.

- Use a self-created homography function to warp the tag into a 8×8 grid (orthographic/upfront view).
- Identify the orientation by locating the white box in the corner of the inner 4×4 grid.
- Decode the Tag ID using the values in the central 2×2 matrix and display it alongside the detected tag.

Code solution for this task should return the corners of the tag as well as its ID and its current orientation.

5.1 Rubric for Task 1 - 30 marks

- Successful Tag detection on given video(s) - 7.5 marks
- Successful Tag detection on webcam stream - 7.5 marks
- Successful Tag identification on given video(s) - 7.5 marks
- Successful Tag identification on webcam stream - 7.5 marks

6 Task 2: 2D Augmented Reality

Superimpose a template image(s) of your choice (e.g. *iitd_logo_template.jpg* is provided) onto the detected AR tags in the video frames.

- Calculate the Homography matrix H between the template image and the tag corners.
- Use a self-created Inverse Homography function to map pixels from the template to the video frame.
- Ensure the image orientation matches the tag's rotation.

6.1 Rubric for Task 2 - 30 marks

- Successful template overlaying on given video(s) - 7.5 marks
- Successful template overlaying on webcam stream - 7.5 marks
- Smooth template overlaying on given video(s) - 7.5 marks
- Smooth template overlaying on webcam stream - 7.5 marks

7 Task 3: 3D Augmented Reality

Develop a 3D Augmented Reality rendering pipeline that projects a 3D model (provided as a .obj file, example. Figure 2) onto a detected AR tag. You can directly read the vertices of the 3D object from .OBJ file and project them using your computed Projection matrix onto the AR Tag. Use of OpenGL or other specialized rendering is not required and not recommended. Boilerplate helper functions are provided for loading and Rendering of OBJ files. Key Steps:

- Recover the 3D camera pose relative to the AR tag. You may refer to the provided supplementary material for computation of projection matrix from Homography in the drive link.
- Anchor the 3D object to the tag's center.
- Match the orientation of the 3D object with AR Tag
- Scaling the 3D model to fit within the tag's physical dimensions.

Note: For Task 3, you will need to calibrate your webcam to get its intrinsics parameters. You may follow the tutorial here: OpenCV camera calibration tutorial.

7.1 Rubric for Task 3 - 20 marks + 10 marks bonus

- Successful 3D object overlaying on given video(s) - 10 marks
- Successful 3D object overlaying on webcam stream - 10 marks
- No flickering and Smooth 3D object overlaying on given video(s) - 5 marks [Bonus]
- No flickering and Smooth 3D object overlaying on webcam stream - 5 marks [Bonus]

For smooth overlaying with no flickering of 3D scene, you are free to experiment with different techniques including Kalman Filter Stabilization, Smoothing Average etc.

8 Implementation Constraints

- You must write your own functions for calculating the Homography, inverse Homography or Projection matrices; do not use any specialized built-in library functions like `cv2.findHomography`, or `cv2.warpPerspective`, or any other image processing function. You may use image read/write and other basic utility functions from OpenCV only. There will be 50% penalty if opencv functions are used for specialized tasks.

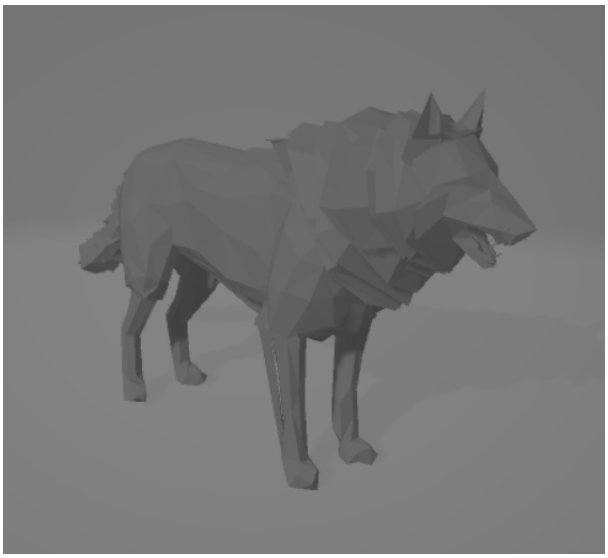

Figure 2: 3D Wolf