CS C280 Computer Vision
Spring 2025 Jitendra Malik, Angjoo Kanazawa

This homework is due on Tuesday, Feb 11, 2025, at 11:59PM.

1 Poor Man's Augmented Reality

In this augmented reality project, you will insert a synthetic object into the scene of **this video**. The basic idea is to use 2D points in the image whose 3D coordinates are known to calibrate the camera for every video frame. Using the camera projection matrix, you will then project the 3D coordinates of a cube onto the image. If the calibration is done correctly, the cube should appear consistently in each frame of the video as seen **here**. Extra resources can be found below [1].

1.1 Keypoints with Known 3D World Coordinates

Start by marking points in the first image of the video using `plt.ginput`, and assign their 3D world coordinates. Use the `imageio` package to read and write videos. We will assume uniform spacing between adjacent coordinates of our box. Assign 3D coordinates by fixing the world coordinate axes at one of the box's corners as seen **here**.

1.2 Propagating Keypoints to Other Frames

To track the keypoints across frames, use OpenCV's tracking API, such as `cv2.TrackerMedianFlow_create()`, initializing separate trackers for each point using an 8x8 patch. The resulting tracklets will look like **this**.

1.3 Calibrating the Camera

Once you have paired 2D image coordinates with their 3D counterparts, fit the camera projection matrix using least squares to map real-world homogeneous coordinates to image coordinates. Perform this calibration for each video frame.

1.4 Projecting a Cube into the Scene

Using the calibrated projection matrix, project 3D points of a cube onto the image. Utilize OpenCV's `draw` function to render the cube in each frame. Scale and translate the cube's coordinates for correct placement. After rendering, compile the processed frames into a video.

1.5 Deliverables

Submit your code as a pdf with comments to title sections 1.1, 1.2, 1.3, 1.4 and a google drive link with the output video of the projected cube.

[1] See **Reference Folder**, **Feature Matching Tutorial**, **Camera Calibration Tutorial**, **Blender**, **Tracking Tutorial**, **Python Rendering Tutorial** for more information.

2 Affine Transforms & Homographies

Here, you will use affine transforms and homographies to map a source image onto several planar surfaces in a target image. As a source image, choose a real photo (e.g., a picture of yourself) as your source image and as a target image, choose a different image that has at least 5 planar surfaces (e.g., billboards, walls, etc.). It is important that the surfaces should have varying normal directions in 3D (e.g., don't map to 5 areas on the same wall). You will map your source image into several planar surfaces in your target image. We also provide a source and target image pair for you to perform the mapping as well.

Given points $\{\mathbf{u}_i\}, \{\mathbf{v}_i\}$, where $\mathbf{u}_i$ is a corner in your source image, and $\mathbf{v}_i$ is the corresponding corner in the target image, both in homogeneous coordinates, we would like to find the transformation $\mathbf{v}_i = T(\mathbf{u}_i)$. The function $T : \mathbb{R}^3 \to \mathbb{R}^3$ is described as follows:

$$\mathbf{v}_i = \begin{bmatrix} v_{ix} \\ v_{iy} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{V_{ix}}{V_{iz}} \\ \frac{V_{iy}}{V_{iz}} \\ 1 \end{bmatrix}, \quad \text{where} \quad \begin{bmatrix} V_{ix} \\ V_{iy} \\ V_{iz} \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{ix} \\ u_{iy} \\ 1 \end{bmatrix} = H\mathbf{u}_i.$$

When solving for H in an affine transform or homography, you may only use basic matrix operations (e.g., multiplication, addition, concatenation, dot products, reshaping, etc.). You may not use a pre-written pseudo-inverse function or packages that solve for the mapping.

2.1 Affine Transform

First, use a 2D affine transform (translation + rotation). Find the least squares solution for H:

$$H^* = \arg\min_H \sum_{i=1}^{4} ||T(\mathbf{u}_i) - \mathbf{v}_i||^2$$

Then, use the transforms to map your chosen image onto the planar surfaces.

To get you started, consider how many degrees of freedom are in an affine transform vs. a homography. Then, consider how to write a matrix equation $A\mathbf{x} = \mathbf{b}$ such that the parameters you want to solve for are in $\mathbf{x}$, and the output $\mathbf{b}$ is what you expect. We provide a **Colab Notebook** with some basic starter code.

In your report, **paste in your filled-out functions** that are mentioned in the deliverables.

Deliverables

(a) Your completed `.ipynb`.

(b) The filled-out function `H = affine_solve(u,v)`, where u, v are $2 \times N$ matrices, representing N corresponding points. The function should return H, a 3×3 matrix.

(c) The filled-out function `v = do_transform(u,H)`, where u is a $2 \times N$ matrix and H is a 3×3 matrix. This function should return v, a $2 \times N$ matrix.

(d) A transformed image with the provided source image (`provided_source.jpg`) mapped onto the provided target image (`sample_target.jpg`) at the coordinates that we provide (`provided_corners.npy`). Save the result as `provided_affine.jpg`. Then, perform the same with your own source image and target image, selecting your own coordinates in the target for the mapping. Save that result as `my_affine.jpg`. Make sure you select a source image which is (roughly) larger than the largest plane in the target image.



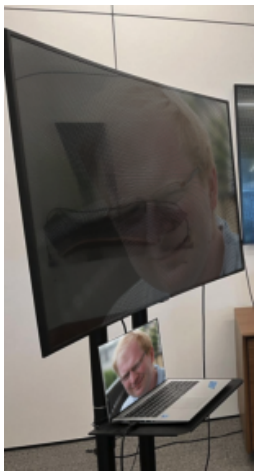

Figure 1: (Left) Provided target image. (Middle) Example homography result with provided source image. (Right) Failure case where provided image is smaller than plane to project onto (think about why).

2.2 Homography

Now, solve for H in a homography, again finding the best transforms to the selected planar surfaces. Once you have written a function to return H, you should reuse your function `do_transform` from the previous question to create the modified target image.

Deliverables

(a) The filled-out function `H = homography_solve(u,v)`, where u, v are $2 \times N$ matrices, representing N corresponding points. The function should return H, a 3×3 matrix.

(b) A homography with the provided source-target pair, saved as `provided_homography.jpg`, as well as your own source-target pair, saved as `my_homography.jpg`.

2.3 Analysis

Answer the following questions in your report:

(a) What constraints are placed on H for the affine transform? How about for the homography?

(b) Is the affine transform able to exactly transform the points from one image to the other? Why or why not?

3 Submission

Submit a single PDF containing written solutions, as well as the notebook (also as a PDF) at the very end. Submit via Gradescope.