CIS 580, Machine Perception, Fall 2025
Homework 2
Due: Wednesday Oct 08 2025, 11:59pm ET
Version of 2025/09/25, 19:06GMT

Instructions

- This is an individual homework and worth 100 points

- Submit your finished files on Gradescope.

- **Start early!** Please post your questions on Ed or come to office hours!

Submission

- You will submit all python files you completed as .py files to Gradescope

- For non-coding portion, please separately submit your solutions, also via Gradescope

Augmented Reality with AprilTags (100pts)

In this section, you will implement a simple augmented reality application. The deliverable is a video that contains several virtual object models as if they exist in the real world. Furthermore, you will be able to specify pixel positions to place an arbitrary object.

You are given a video with an AprilTags (https://april.eecs.umich.edu/software/apriltag) in each frame. These tags are usually used in robotics for determining the pose of the camera. To make things easier, we provide the 4 corners coordinates (in pixel) as well as the size of the tags.

Your main task in this problem is to recover the camera poses with two different approaches: 1) solving the Perspective-N-Point (PnP) problem with coplanar assumption and 2) solving the Persepective-three-point (P3P) and the Procrustes problem.

After retrieving the 3D relationship between the camera and world, we can place an arbitrary objects in the scene.
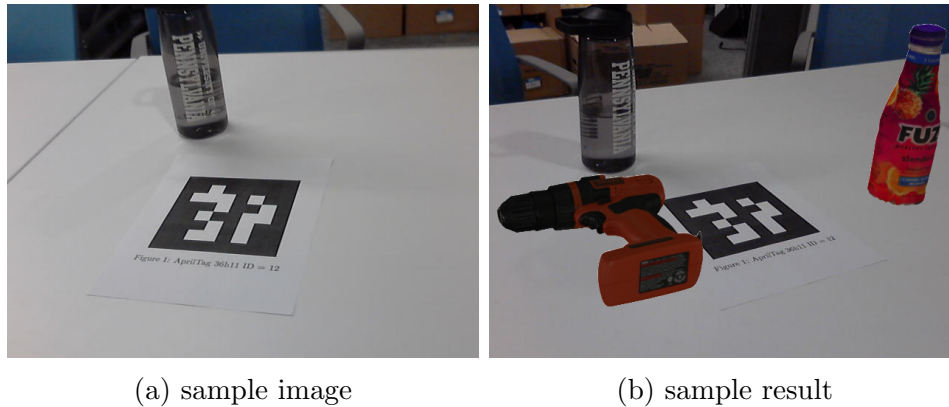

(a) sample image   (b) sample result

Figure 1: Projecting objects on the table
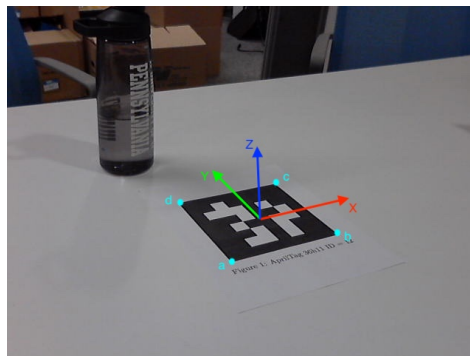
1 Establish World Coordinate System (5 pts)

Figure 2: world coordinate setup

To know where the camera is in the world, we must first define a world coordinate system. We conveniently place the world coordinate at the center of the tag as shown in Figure 2.

From the lecture, we know that recovering camera pose from 2D-3D correspondence requires at least 3 points. We provide all 4 AprilTag corners in **corners.npy**. Theses corner coordinates will be used in **main.py**.

Let a, b, c and d be the four corners in the image. Your first task is to find the 3D world coordinates of a, b, c and d by completing **est_Pw.py** given the length of the side of the tag. We define the world coordinate system to be centered at the center of the AprilTag, with x, y direction pointing towards edge bc and cd. The z direction is

oriented vertically up out of the tag. All tag points are assumed to be coplanar. Please refer to Figure 2 for the correct world coordinate configuration.

2 Solve PnP with Coplanar Assumption (25 pts)

In this section, you will estimate the camera pose from an AprilTag based on homography estimation. Recall that a homography is a projective transformation between planes, which you have implemented in hw1. The technique for solving camera pose follows closely the lecture slides "Extrinsics from Collineation - Pose from Projective Transformations"

Since the world frame is conveniently placed on the tag, the z component of the corners should be zero. Following the slides, we have

$$
\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \sim K \begin{pmatrix} r_1, r_2, T \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}
$$

where K $\left(r_1, r_2, T\right)$ is a homography H. To recover H, feel free to re-use **est_homography.py** that you implemented in hw1. We also provide a reference solution.

After obtaining H, you should follow the slides to recover R and t. Note that in the slide, $R = R_w^c$ and $t = t_w^c$ whereas **est_homography.py** here should return R_c^w and t_c^w, which describe camera pose in the world. You should complete the function in **solve_pnp.py**

3 Solving Perspective-Three-Point problem (30 pts)

Here you will calculate the camera pose by first calculating the 3D coordinates of any 3 (out of 4) corners of the AprilTag in the camera frame. In Part 1, the 3D coordinates of the same points in the world frame have been calculated. You will use this correspondence to solve for camera pose R_c^w and t_c^w in the world frame by implementing solving the Procrustes problem.

3.1 P3P Derivation
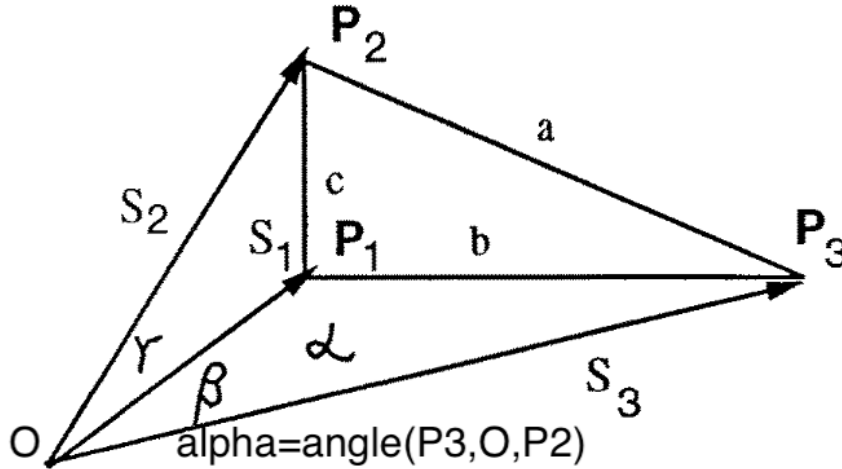

Figure 3: Illustrates the geometry of the three point space resection problem. The triangle side lengths a, b, c are known and the angles α, β, γ are known (α is the angle between $\mathbf{P}_2$ and $\mathbf{P}_3$, β is the angle between $\mathbf{P}_1$ and $\mathbf{P}_3$ and γ is the angle between $\mathbf{P}_1$ and $\mathbf{P}_2$). The problem is to determine the lengths s_1, s_2, s_3 from which the 3D vertex point positions (in the camera frame) P_1, P_2, P_3 can be immediately determined.

You only need to solve up to a 4th degree polynomial, i.e. estimating the coefficients of this polynomial in terms of $a, b, c, \alpha, \beta, \gamma$. We code them up in the next part to get the roots of the polynomial.

Part of the proof for P3P is given in class (PnP slides). Please refer to this document for more details in solving P3P. P3P reference paper. You need to use the equations from Grunert's method from this paper. This is an exercise in absorbing a paper and coding it.

3.2 P3P (Programming)

Please complete function in **solve-p3p.py**. Important Points to consider:

- you have 4 corners and are free to use any 3 of them.

- You can use numpy.roots to get the roots of the polynomial

- You will get up to 4 roots for the polynomial. The remaining point should be used to select the best R and t.

4

4   Solving Procrustes Problem (20 pts)

You need to use the correspondence of the same 3 points in the world frame and the camera frame and solve for camera pose using the Procrustes method. Use the Procrustes slides from class for your reference.

$$\min_{R,T} \sum_i^{N=3} ||A_i - RB_i + T||^2$$

5   Pick Placing Location (15 pts)

Now that you have R and T between the camera and the world coordinate, you can place an object at any location on the table! However, there is one piece missing from interactively selecting a place on the table from a picture - we need to be able to click on a pixel and know its coordinate in the world.

Suppose you have pick a pixel on the table at (u, v), you can find out the 3D location of that point with respect to the world coordinates. Given R_c^w and T_c^w and the camera intrinsic matrix K, please write down mathematically how to calculate the 3D location $X_w \in R^3$ in world coordinates corresponding to pixel $\hat{x} = [u, v, 1]^T$. (**Hint**: points on the table follow $z = 0$) You can use letter and subscripts to represent entries in the matrix, e.g. A_{ij} represents the value at ith row and jth column of A matrix.

Programming: Please complete **est_pixel_world.py**.

6   Put Everything Together (5 pts)

At this point, you have all the components needed for simplified augmented reality application based on AprilTags. In this section, you will use the completed functions to generate a video. We provide a series of frames in the **data** folder. For each frame, you are place an arbitrary number of objects in the scene. Now you can place your objects on the table by executing main.py. We have picked two pixel locations in the file. These coordinates correspond to the centers of the virtual objects in the first frame. You should use these coordinates in your submitted video. For each frame, keep the objects static in the world coordinate and render the image from the current camera view. At the end, you are asked to combine these frames into a **.gif** file. **You should submit the .gif file to Gradescope with the code**.

If you want to use your own points to project, please add **--click_points** option when running **main.py**. You will be prompt to click two points on the screen, these points

will be selected as the centers of the two objects. Feel free to swap the objects with your favourite models and submit the gif along!

Files to complete and submit

1. **est_Pw.py**
   This function is responsible for finding world coordinates of a, b, c and d given tag size

2. **solve_pnp.py**
   This function is responsible for recovering R and t from 2D-3D correspondence with coplanar assumption

3. **est_pixel_world.py**
   This function is responsible for solving 3D locations of a pixel on the table.

4. **solve_p3p.py**
   This file has two functions P3P and Procrustes that you need to write. P3P solves the polynomial and calculates the distances of the 3 points from the camera and then uses the corresponding coordinates in the camera frame and the world frame. You need to call Procrustes inside P3P to return R,t.

5. **VR_res.gif**
   This file is generated automatically by executing main.py and needs to be submitted with other ∗.py file.