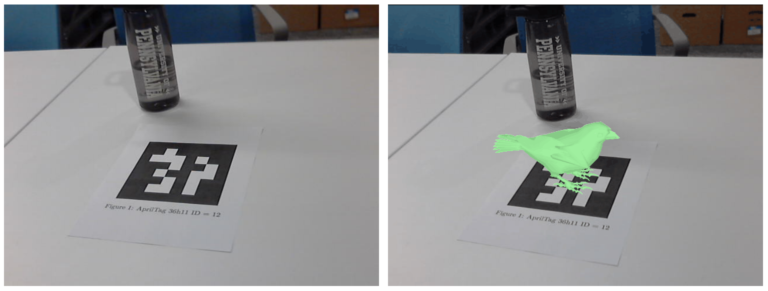
CIS 580: Machine Perception, Spring 2020
Homework 5
Due: February 28, 2020 at 11:59pm

Instructions

- This is an individual homework.
- You must submit your solutions on Gradescope, the entry code is MB8ZJP. We recommend that you use LaTeX, but we will accept scanned solutions as well.
- You need to submit a report which contains the solution to Section 3.1 and a short explanation for all the other Sections.
- You need to upload the .py files that you have edited along with the generated .gif files.
- Please complete this homework with Python 3. To run this homework successfully, you will need the following python packages: **numpy, imageio, trimesh, pyrender**.
  You are not allowed to use cv2 for this assignment.
- Start early! If you get stuck, please post your questions on Piazza or come to office hours!

Introduction to Augmented Reality using AprilTags

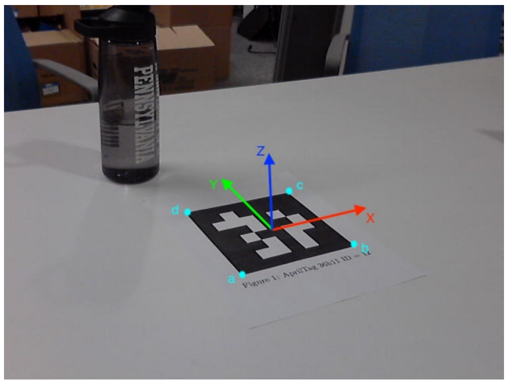
In this assignment, you are given a set of images with an AprilTag. You are also given the size of the tag and the pixel locations of its corners on the images. Your task is to recover the camera pose **求 R, T** with two different approaches: PnP with coplanar assumption, and P3P followed by Procrustes. After you recover the camera pose, we will place a virtual camera at that same position to render a virtual bird as if it stands on the tag.



(a) sample image (b) sample result

Figure 1: Projecting a bird mesh on the AprilTag center

1 Problem 1 - 5pts, Establish a World Coordinate



(a) world coordinate setup

To know where the camera is in the world, we must first define a world coordinate system. For this assignment, since we want to place a bird onto the tag, it is convenient to place the world coodinate at the center of the tag as shown in the figure above.

From the lecture, we know that to recover camera pose from 2D-3D correspondence, we need at least 3 points. Here the AprilTag provides us 4 points. We used the detection software for AprilTag to find the 2D pixel coordinates of a, b, c, d, which have been provided to you in **corners.npy** and will be used in **run_PnP.py** and **run_P3P.py**.

Your first task is to find the 3D world coordinates of a, b, c, d by completing **est_Pw.py** given the length of the side of the tag. With the world coordinate placed at the center of the tag, this should not be difficult. This function will be used for the next two sections.

1.1 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 Problem 2 - 25pts, Solve PnP with Coplanar assumption

After getting the 2D-3D correspondence, we will recover the camera pose with two different approaches. This section covers the first method which closely follows 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

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \sim K \left(r1, r2, T\right) \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

where K $\left(r1, r2, T\right)$ is a homography H. To recover H, you will use **est_homography.py** that you've programmed in hw2. We also provided a completed version. You are free to choose either.

After getting H, you should follow the slides to recover R and t. Note that in the slide, $R = R_{cw}$ and $t = t_{cw}$, but the function requires to return R_{wc} and t_{wc}, which describe camera pose in the world. You will complete all of above inside **PnP.py**

2.1 Visualization

Once you complete **est_Pw.py** and **PnP.py**, run **python run_PnP.py** in terminal. This will generate a GIF called bird_collineation.gif.

2.2 Files to complete and submit

1. **PnP.py**
   This function is responsible for recovering R and t from 2D-3D correspondence
2. **bird_collineation.gif**
   This GIF visualizes the bird for method 1.

3 Problem 3 - 70pts, Solving the Perspective 3-Point problem

Here you will go through the process of calculating the camera pose by first calculating the 3D coordinates of any 3 (out of 4) corners of the AprilTag in the camera frame. You already know the 3D coordinates of the same points in the world frame from Part 1, so you will use this correspondence to solve for Camera Pose R, t in the world frame by implementing Procrustes.

3.1 P3P Derivation 20 pts

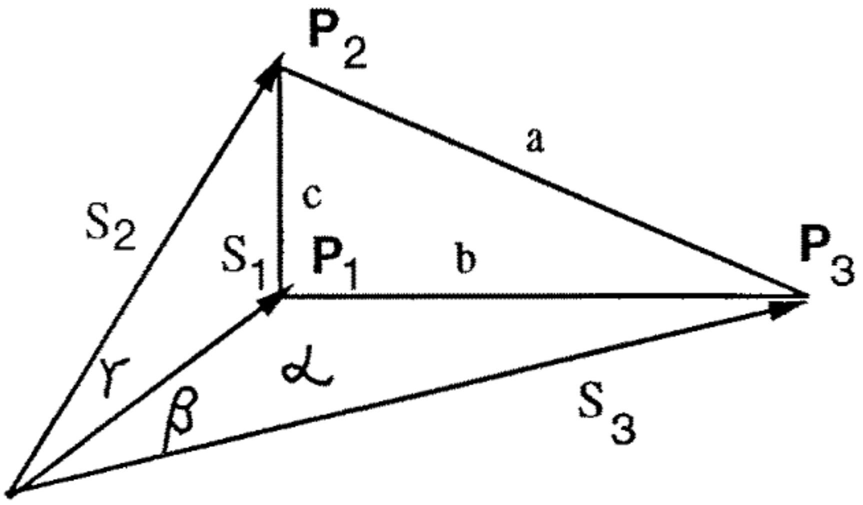


Figure 3: Illustrates the geometry of the three point space resection problem. The triangle side lengths a, b, c are known and the angles α, β, γ are known. The problem is to determine the lengths s_1, s_2, s_3 from which the 3D vertex point positions (in the camera frame) P_1, P_2, P_3 can be immediately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

You only need to solve till you get a 4th degree polynomial. Basically get the coefficients of this polynomial in terms of $a, b, c, \alpha, \beta, \gamma$, then we will code them up in the next part to get the roots of the polynomial.

You already have a part of the proof for P3P in class (PNP slides). You can refer this document if you face difficulty in solving P3P. P3P reference paper (you need to use Grunert's solution)

3.2 Coding Sections

3.2.1 P3P - 25 pts

In this function, you need to implement P3P to basically calculate the 3D coordinates of the 3 points in the camera frame. Refer slides "Perspective N Point Problem"
You will define a, b, c and α, β, γ and then compute the coefficients of the polynomial you got in the previous part. Then you will solve the polynomial to get its roots which will then give you the distances $s1, s2, s3$. You will then use these distances to get the 3D coordinates of the 3 points in the camera frame.

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 total of 4 roots out of which 2 will be real, you need to select the one that gives all the distances of the 3 points from the camera to be positive.

3.2.2 Procrustes - 25 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.

就酱?

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

3.3 Visualization

Once you complete **P3P.py**, run **python run_P3P.py** in terminal. This will generate a GIF called bird_P3P.gif.

3.3.1 Files to complete and submit

1. **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.

2. **bird_P3P.gif**
   This GIF visualizes the bird for method 2.