CS 498: Assignment 1: Perspective Projection

Hongqing Liu, hl85

February 7, 2023

Submission

In this assignment you will be modifying the files homography.py and perspective.py. Please put together a single PDF with your answers and figures for each problem, and submit to Gradescope (Course Code: BBX6NE). We recommend you to add your answers to the latex template files we provided. For code submission, make sure you use the provided ".py" files with your modification and the dumped ".npz" file. The graders will check both your PDF submission and your code submission if needed.

Homography [8pts]

In this question, we will examine properties of homography transformations and see how to estimate a planar homography from a set of correspondences.

Question 1 [1pt]: You are given a photo of the State Farm Center (`uiuc.png`), UIUC's indoor arena that hosts our basketball teams. We marked and plotted the four corners of the court, as shown in Fig. 1 (left). A standard basketball court is 28.65 meters long and 15.24 meters wide, as shown in Fig. 1 (right). Now let's consider a 2D coordinate system defined on the planar surface of the basketball court. The origin of this coordinate system is point #3; the long edge is x axis and short edge is y axis. Please write down the four corner's coordinates in this 2D coordinate frame (in meters) and fill in your answer in the numpy array "corners_court".

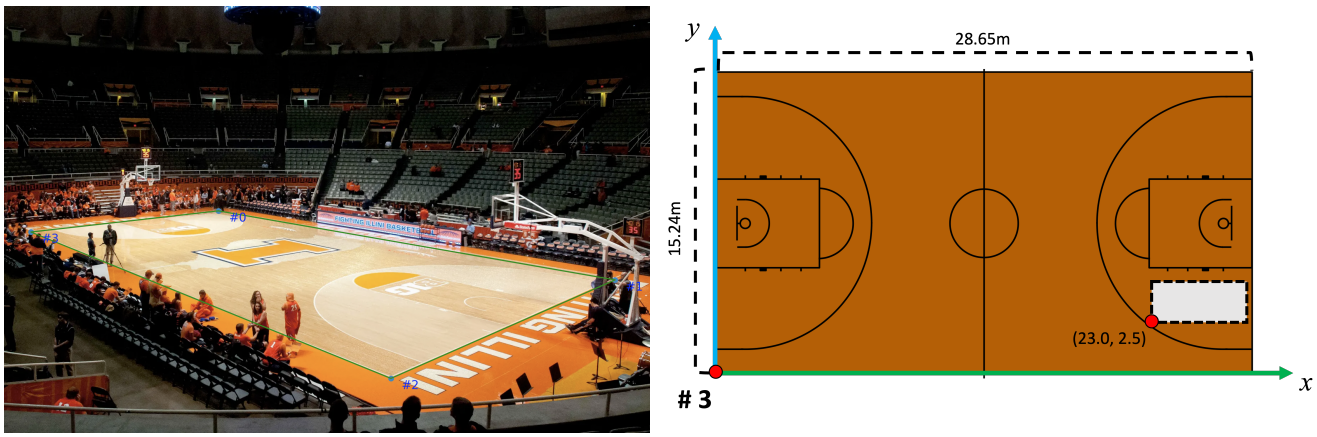


Figure 1: Left: target image with keypoints; Right: court dimensions and coordinate convention.

Answer: The four coner's coordinates are point0$(0, 15.24)$, point1$(28.65, 15.24)$, point2$(28.65, 0)$, point3$(0, 0)$.

$$conersCourt = \begin{bmatrix} 0 & 15.24 \\ 28.65 & 15.24 \\ 28.65 & 0 \\ 0 & 0 \end{bmatrix}$$

Question 2 [3pts]: Given your Q1's answer, now we could establish four pairs of point correspondence between two planes: the 2D basketball court plane and the 2D image plane. Using what we have learned in Lecture 3, complete the function `findHomography` and use it to estimate the homography matrix from the court to the image. Please briefly explain what you do for homography estimation and report the resulting matrix in this document. Hints: 1) you might find the functions `vstack, hstack` to be handy for getting homogenous coordinates; 2) you might find `numpy.linalg.svd` to be useful; 3) lecture 3 described the homography estimation process.

Answer:

Since we want to project a 3D object into a 2D image, we can transform the 3D coordinate into 2D by a projection matrix. Here we use homogeneous world coordinates(4x1) and homogeneous image coordinates(3x1), as shown below. The last item in image coordinates z shows the scalar ambiguity.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Since the basketball court plane is in a 2D coordinate, which means it is a planar object and its z-coordinate is always 0. In that case, p_3, p_7 and p_11 has no attribute to xyz 2D image coordinate.

Since we know that x and PX have same orientation, their cross product will equal 0, as shown below:

$$x = kPX \rightarrow \vec{x} \times \vec{PX} = 0 \tag{1}$$

Each pairs of corresponding points can establish 2 equations, since the P[3,3] needs to be 1, we need a eight-simultaneous equations to find the solution, which means we need four pairs of corresponding points. The equation systems can be wirtten in the form of matrix multiplication, as follows.

$$\begin{bmatrix} 0 & 0 & 0 & X_1 & Y_1 & 1 & -y'_1X_1 & -y'_1Y_1 & -y'_1 \\ X_1 & Y_1 & 1 & 0 & 0 & 0 & -x'_1X_1 & -x'_1Y_1 & -x'_1 \\ 0 & 0 & 0 & X_2 & Y_2 & 1 & -y'_2X_2 & -y'_2Y_2 & -y'_2 \\ X_2 & Y_2 & 1 & 0 & 0 & 0 & -x'_2X_2 & -x'_2Y_2 & -x'_2 \\ 0 & 0 & 0 & X_3 & Y_3 & 1 & -y'_3X_3 & -y'_3Y_3 & -y'_3 \\ X_3 & Y_3 & 1 & 0 & 0 & 0 & -x'_3X_3 & -x'_3Y_3 & -x'_3 \\ 0 & 0 & 0 & X_4 & Y_4 & 1 & -y'_4X_4 & -y'_4Y_4 & -y'_4 \\ X_4 & Y_4 & 1 & 0 & 0 & 0 & -x'_4X_4 & -x'_4Y_4 & -x'_4 \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \mathbf{0}$$

After that, I form a matrix A from the set of correspondences, each row of the matrix corresponds to a single correspondence. After that, I perform SVD on the matrix A to obtain its singular values and singular vectors. The last column of the right singular vector corresponding to the smallest singular value represents the entries of the projection matrix. After that, I resized it into 3x4 and normalized the projection matrix by divde it by its last element.

The projection matrix is shown as follows:

$$P = \begin{bmatrix} 1.20809640e+01 & 4.44696370e+01 & 7.92073117e+01 \\ -8.05770293e+00 & 5.76174339e+00 & 6.42252451e+02 \\ -2.12530603e-02 & 1.65326163e-02 & 1.00000000e+00 \end{bmatrix}$$

Question 3 [4pts]: We want to promote our CS498 class across the university. One of the marketing ideas we came up is to create an on-court ad in Illinois's State Farm Center. Now you are taking in charge of the virtual signage task – inserting the logo of our class (`logo.png`) electronically onto the basketball court (`court.png`). Specifically, the size of the logo needs to be 3x6 meters, and we want to place the bottom left logo corner at $(23.0, 2.5)$ on the basketball court. In order to do so, we need two steps:

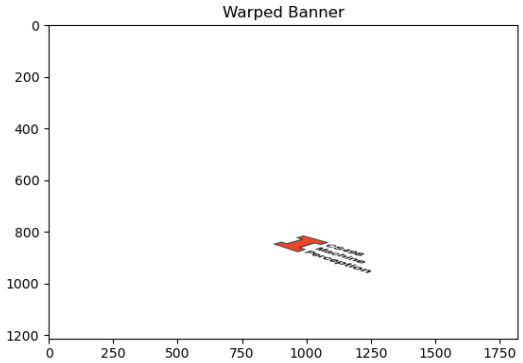
- **3.a [1pt]:** calculate the homography transform between the image coordinate of the two images `logo.png` and `court.png`. Hints: 1) could you compute the transform from `logo.png` to the basketball court 2) could you leverage the homography matrix you computed in Question 2?

  Answer: the homography transfor, between image coordinate of two images can be cauculate by dot product of transformation matrixs between court cooridnates image, court 3-d image and logo image, court coordinates image. The transformation matrix is shown below:

$$targettransform = \begin{bmatrix} 7.24857838e-02 & -2.66817822e-01 & 6.01652487e+02 \\ -4.83462176e-02 & -3.45704603e-02 & 4.88614872e+02 \\ -1.27518362e-04 & -9.91956978e-05 & 6.02109002e-01 \end{bmatrix}$$

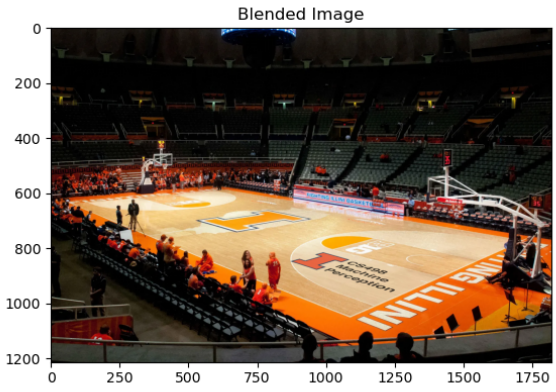
- **3.b [2pt]:** complete the `warpImage` function and use the homography you computed in 3.a to warp the image `logo.png` from its image coordinate to the `court.png`'s image coordinate. Hints: 1) suppose $(x', y', 1)^T = \mathbf{H}(x, y, 1)^T$, we have $I_{\text{target}}(x', y') = I_{\text{source}}(x, y)$; 2) you might find `numpy.meshgrid` and `numpy.ravel_multi_index` to be useful.

  Answer: I have get rid of any loop over all pixels, the result of warp image is shown below.



- **3.c [1pt]:** alpha-blend the warped logo onto the court: $I = \alpha F + (1 - \alpha)B$

  Answer: Here I set alpha logo to be 0.8 and use the following codes to solve the problem of uneven transparency.




```python
# Create the output image

warped_img = original_img

warped_img_1 = np.zeros(shape_=(shape[0],shape[1],4), dtype=img.dtype)
warped_img_1[y, x] = img[indices[1], indices[0]] * alpha_logo + warped_img[y, x] * (1 - alpha_logo)

warped_img[:,:,0] = np.where([warped_img_1[:, :, 3] != 1], warped_img[:,:,0], warped_img_1[:,:,0])
warped_img[:,:,1] = np.where([warped_img_1[:, :, 3] != 1], warped_img[:,:,1], warped_img_1[:,:,1])
warped_img[:,:,2] = np.where([warped_img_1[:, :, 3] != 1], warped_img[:,:,2], warped_img_1[:,:,2])
warped_img[:,:,3] = np.where([warped_img_1[:, :, 3] != 1], warped_img[:,:,3], warped_img_1[:,:,3])
```

Perspective Projection [7pts]

Till now we have been working on transformations between 2D coordinate systems. Now let us to lift up to the 3D coordinate. Consider the same image of the state farm center, but this time, we annotate four additional points, corresponding to the four corners of the basketball backboard.

Question 4 [1pt]: The lower rim of the backboard is 2.745 meters above the ground; the backboard width is 1.83 meters and backboard height is 1.22 meters. The backboard protrudes 1.22 meters out from the baseline. Now let us consider the world frame. The world frame coordinate origin is at the point #3, where x-axis is along the long edge (sideline) and y-axis is along the short edge (baseline), and z-axis is upwards, perpendicular to the basketball court. Could you compute the 3D coordinates of all the eight keypoints?

Answer: The 3D coordinates of all eight keypoints in non-homogeneous are shows as follows, each row represents a keypoint:

$$eightpoints = \begin{bmatrix} 0 & courtwidth & 0 \\ courtlength & courtwidth & 0 \\ courtlength & 0 & 0 \\ 0 & 0 & 0 \\ boardtobaseline & (courtwidth - backboardwidth)/2 & lowerrim + backboardheight \\ boardtobaseline & (courtwidth + backboardwidth)/2 & lowerrim + backboardheight \\ boardtobaseline & (courtwidth + backboardwidth)/2 & lowerrim \\ boardtobaseline & (courtwidth - backboardwidth)/2 & lowerrim \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 15.24 & 0 \\ 28.65 & 15.24 & 0 \\ 28.65 & 0 & 0 \\ 0 & 0 & 0 \\ 1.22 & 6.705 & 3.965 \\ 1.22 & 8.535 & 3.965 \\ 1.22 & 8.535 & 2.7449999999999997 \\ 1.22 & 6.705 & 2.7449999999999997 \end{bmatrix}$$

Question 5 [4pt]: Now we have established eight pairs of 2D and 3D keypoint correspondences. Consider this set of correspondence pairs in homogeneous coordinates $(\mathbf{x}_i, \mathbf{X}_i)$. Our goal is to compute a perspective projection matrix such that:

$$\mathbf{x}_i = \alpha \mathbf{P} \mathbf{X}_i = \alpha \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \mathbf{p}_3^T \end{bmatrix} \mathbf{X}_i = \alpha \begin{bmatrix} \mathbf{p}_1^T \mathbf{X}_i \\ \mathbf{p}_2^T \mathbf{X}_i \\ \mathbf{p}_3^T \mathbf{X}_i \end{bmatrix}$$

where $\mathbf{p}_i^T$ is the ith row of $\mathbf{P}$; $\mathbf{x}_i = (x_i, y_i, 1)^T$; α is an arbitrary scalar.

- **5.a [1pt]:** Please prove that:

$$\mathbf{A}_i \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix} = \mathbf{0} \text{ where } \mathbf{A}_i = \begin{bmatrix} \mathbf{0} & \mathbf{X}_i^T & -y_i \mathbf{X}_i^T \\ \mathbf{X}_i^T & \mathbf{0} & -x_i \mathbf{X}_i^T \end{bmatrix}.$$

Answer: Since we know that

$$\mathbf{x}_i = \alpha \mathbf{P} \mathbf{X}_i$$

α here represents scalar ambiguity, so we can put it into x,y coordinate and write it as

$$\mathbf{x}_i^\star = \mathbf{P} \mathbf{X}_i$$

which can be written as follows:

$$\begin{bmatrix} \frac{x}{\alpha} \\ \frac{y}{\alpha} \\ \frac{z}{\alpha} \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

which can be written as follow after normalized:

$$x' = \frac{h_1 X + h_2 Y + h_3}{h_7 X + h_8 Y + h_9} \quad y' = \frac{h_4 X + h_5 Y + h_6}{h_7 X + h_8 Y + h_9}$$

$$h_1 X + h_2 Y + h_3 - x' (h_7 X + h_8 Y + h_9) = 0$$
$$h_4 X + h_5 Y + h_7 - y' (h_7 X + h_8 Y + h_9) = 0$$

which can be written into matrix multiplication form as follows:

$$\begin{bmatrix} 0 & 0 & 0 & X & Y & 1 & -y'X & -y'Y & -y' \\ X & Y & 1 & 0 & 0 & 0 & -x'X & -x'Y & -x' \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ \cdots \\ h_9 \end{bmatrix} = 0$$

which is equal to:

$$\mathbf{A}_i \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix} = \mathbf{0} \text{ where } \mathbf{A}_i = \begin{bmatrix} \mathbf{0} & \mathbf{X}_i^T & -y_i \mathbf{X}_i^T \\ \mathbf{X}_i^T & \mathbf{0} & -x_i \mathbf{X}_i^T \end{bmatrix}.$$

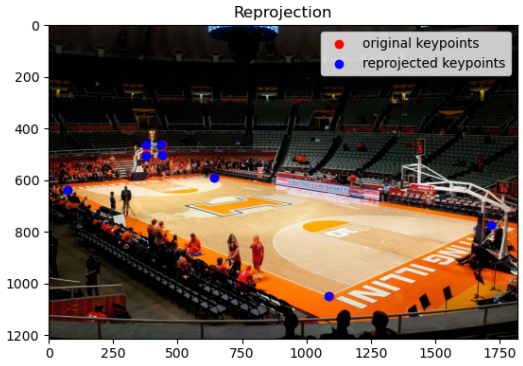
- **5.b [3pt]:** complete the `findProjection` function and use this function to recover the perspective camera projection matrix, from the 3D world coordinate to the 2D image coordinate. Hints: 1) establish a linear system based on your derivation 2) given the eight keypoint correspondences, how many equations and how many unknown variables? 3) consider using `numpy.linalg.svd` to solve the linear system. 4) you might find `numpy.concatenate` and `numpy.reshape` to be handy.

  Answer: The projection matrix is shown as follows:

  projection matrix =

$$\begin{bmatrix} 1.18682798e+01 & 5.03681893e+01 & 1.45686779e+01 & 7.34119776e+01 \\ -8.26855706e+00 & 8.76915144e+00 & -2.39149474e+01 & 6.35921983e+02 \\ -2.16285953e-02 & 2.02166814e-02 & 4.33368607e-02 & 1.00000000e+00 \end{bmatrix}$$

  And the reporjection result is shown as follows:



- **5.c [1pt bonus]:** Try to answer the following questions: 1) assuming perfect correspondences, could we recover the projection matrix using even fewer points? Please indicate how many correspondences at least we need. 2) could we just use correspondences on the ground plane to recover this projection matrix? Please explain why or why not.

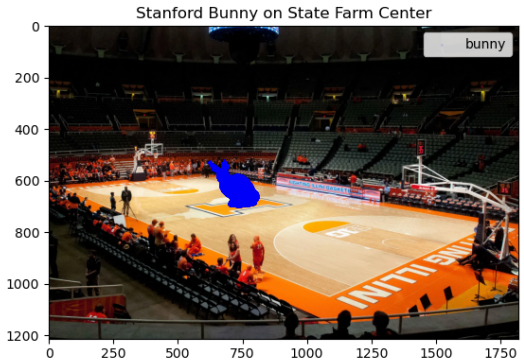
  Answer: 1) The projection matrix needs at least 6 points to recover. This is because with 6 correspondences, we can construct a system of 12 equations and solve for the 12 unknown variables in the projection matrix. Any fewer correspondences would result in an underdetermined system.

  2) we cannot just use correspondences on the ground plane to recover the projection matrix. This is because we need data from all three dimensions to recover the projection matrix. To be specific, if we only have correspondences on the groud plane, its z-value will be 0, which means we can not determine the value of p_3, p_7 and p_11 in the projection matrix.

Question 6 [2pt]: Using our estimated projection matrix in Q5, we could insert any 3D object in the world coordinate frame onto the 2D image. In this question, let's consider the a simple case where we want to project a 3D point cloud onto the image plane and plot them. Suppose we want to put a giant stanford bunny statue in the center of the basketball court, could you compute the 2D coordinate of every vertex of this bunny using the projection matrix? Hints: 1) you might need to transform the bunny to the court center first 2) make sure the bunny is above the ground.

Answer: Firstly I centralize the 3D object in its x-axis and y-axis to 0 and add half of the court length and half of the court width to first channel and second channel (x direction and y direction) of each verts respectively to put the bunny to the center of the court and minus each pixels' third channel value with the smallest value for all pixels in the third channel (z-direction) to put the bunny above the ground.Then I use projection matrix to project 3-d points into 2-d image.

The projection result of stanford bunny statue is shown as follows:



Question 7 [2pt bonus]: Try to use an off-the-shelf renderer to render the 3D mesh in a more realistic manner (e.g. pyrender, moderngl, blender, mitsuba2). You could consider the following aspects that might influence the realism: 1) shading 2) shadows 3) occlusion reasoning 4) lighting environments.

Hints: for some libraries, you probably need to decompose the projection matrix into valid intrinsic and extrinsic matrices. You can also manually annotate additional key points to make the projection matrix estimation more accurate in this experiment.

Answer: I use pyrender to render the 3D mesh. Firstly I centralize the 3D points in x and y direction, then I use mesh.frompoints to transform 3D coordinates into mesh items. After that, I set scene, light parameter and choose orthographic camera. The position of light and scene is same to mesh coordinate. The camera coordinate system is rotated firstly around x-axis by $\pi/4$ and then around z-aixs by $\pi/4$. The result is shown as follow, The left one is generated by only vertices and the right one is generate by both vertices and faces: