1 Conventions

Coordinates in the world frame will be denoted as (U, V, W). Coordinates in the camera frame will be denoted as (X, Y, Z). Coordinates in the 2D image plane will be denoted as (x, y). Pixel coordinates will be denoted as (u, v).

The court has the following dimensions and coordinate system. Note that we are centering the world coordinate system at the center of the court and that the w axis points towards the ceiling.

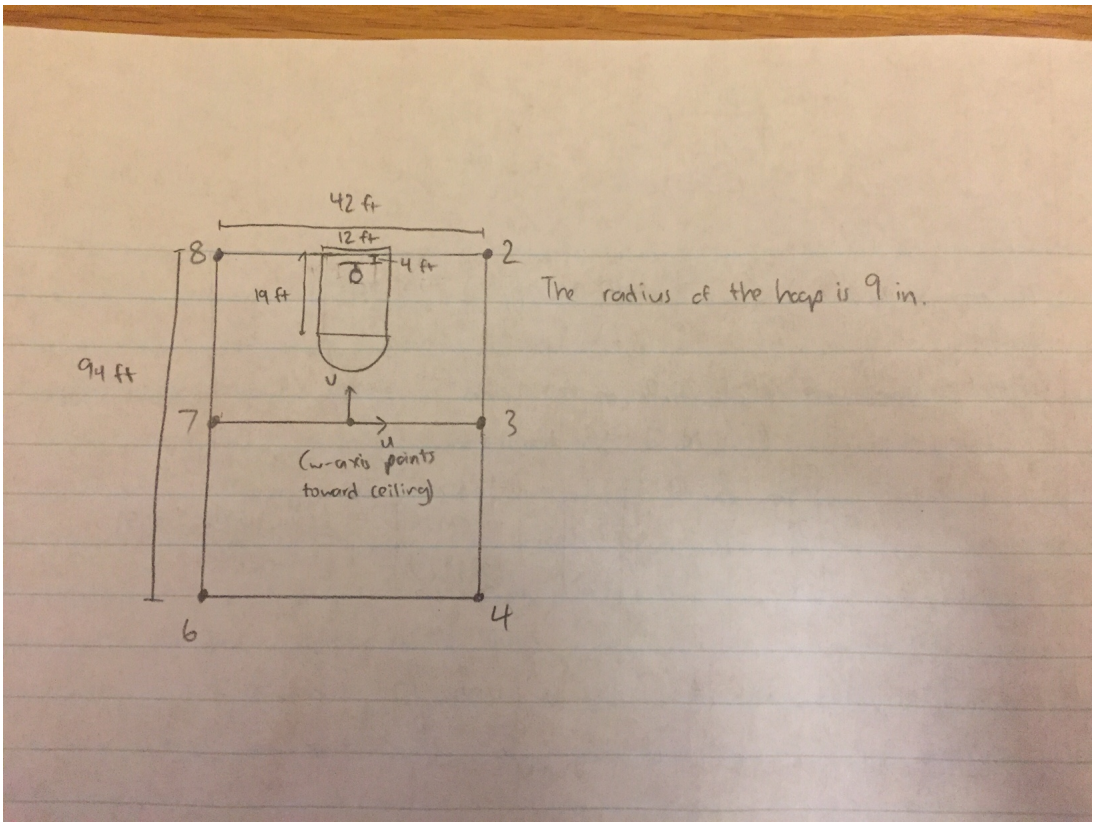

Figure 1: A diagram of the MAC basketball court with dimensions

For the camera frame, we will adopt the convention that the Z axis points in the direction that the camera is pointing. See the following diagram, where the rectangular prism represents the camera:

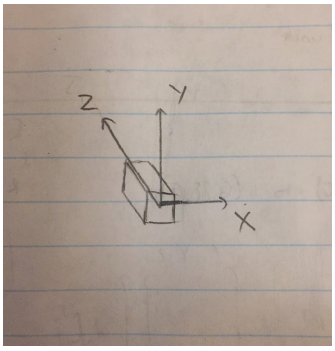

Figure 2: A poorly drawn representation of the camera coordinate system

The image coordinate system has the origin at the center of the image, with the x-axis pointing to the right and the y-axis pointing up (i.e. standard conventions). The pixel coordinate system has $(0, 0)$ representing the pixel in the top left corner. Also note that it is represented as an array in Python with the first coordinate representing the row of the pixel (so the u axis points down and corresponds to the image $-y$ axis) and the second coordinate representing the column (so the v axis points to the right and corresponds to the image x axis).

2 World to Camera

To get from world coordinates to camera coordinates, we must perform a translation and then a rotation. Let the camera be placed at location $\boldsymbol{C}$ (denoting a vector) in the world frame, and let the rotation matrix be R. Since the rotation matrix R is designed to go from the world to the camera coordinate system, R^T is designed to go from the camera to the world. Thus, the first column of R^T should represent the camera X-axis in world coordinates, the second column should represent the Y-axis, and the third column should represent the Z-axis. In terms of R, we see that the first *row* of R must be equivalent to the camera X-axis in world coordinates, etc.

In practical terms, it is relatively easy to estimate what the camera Z-axis is in world coordinates. By looking at one frame and identifying the center pixel (I used MS Paint for this), we can identify where in the world coordinate system the camera is pointing. The easiest way to do this is to consider where the ray pointing out the center of the camera hits a solid object. For example, if the center pixel is on the rim or the backboard, we can write a vector representing where that point is in relation to the camera position. We then normalize the vector.

To get the X-axis, we rely on the assumption that the camera X-axis has a W-coordinate of 0 in the world frame (i.e. that it is parallel to the ground). This is a reasonable assumption because

we start with the camera pointed parallel to the ground. When we tilt the camera slightly upwards, we don't rotate it in any other way, so the X-axis stays stationary while the Z and Y axes rotate (see Figure below). Thus, we look at the first two world coordinates of the camera Z-axis, flip the order of the coordinates, and change one sign (we flip the U-coordinate so that the X axis points to the right). This ensures that the X and Z axes are orthogonal (have a dot product of 0).

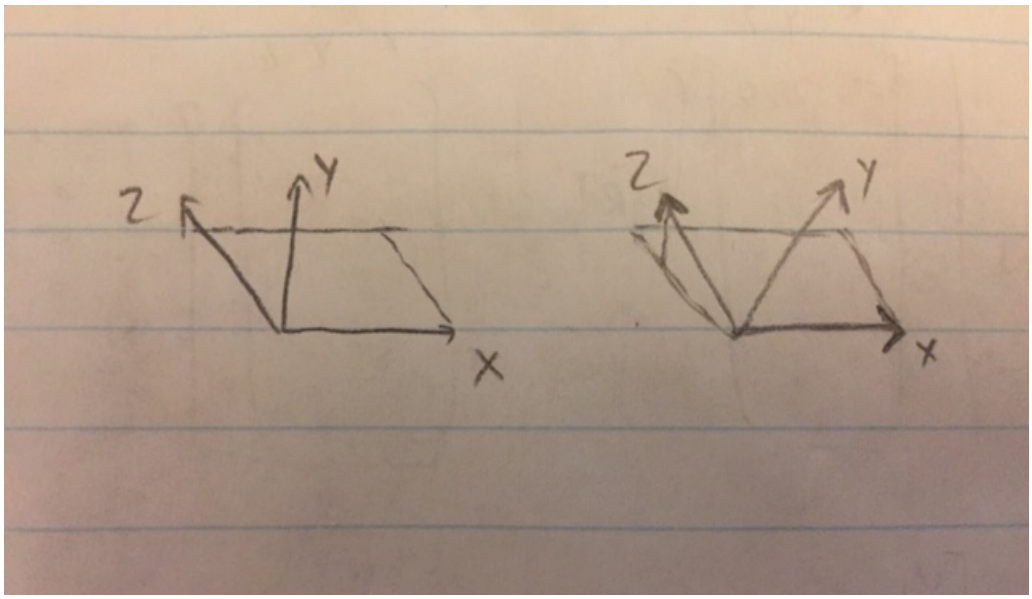

Figure 3: Why the camera X axis is parallel to the ground

Finally, to get the Y-axis, we first normalize the X and Z axes, and then take the cross product of X and Z. We make sure X comes first so that the Y-axis points in the right direction by the right hand rule. We put our three unit vectors together and we have a rotation matrix R! We can now relate camera coordinates to world coordinates:

$$\boldsymbol{P}_C = R(\boldsymbol{P}_W - \boldsymbol{C})$$

3 Camera to Image

The key idea is that the "image plane" is an imaginary plane f units down the Z-axis of the camera, where f is the focal length. To get the image plane coordinates, we use similar triangles and scale

down a point that is Z units away to a plane that is f units away. We get:

$$x = \frac{f}{Z}X$$
$$y = \frac{f}{Z}Y$$

Note: Some sources like to use "homogeneous coordinates" so that we can write a matrix equation $\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$. Then $(x', y', z') = (Xf, Yf, Z)$ and $(x, y) = (\frac{x'}{z'}, \frac{y'}{z'})$. I don't really understand the need for this but if one of you guys wants to look into it more, please go ahead.

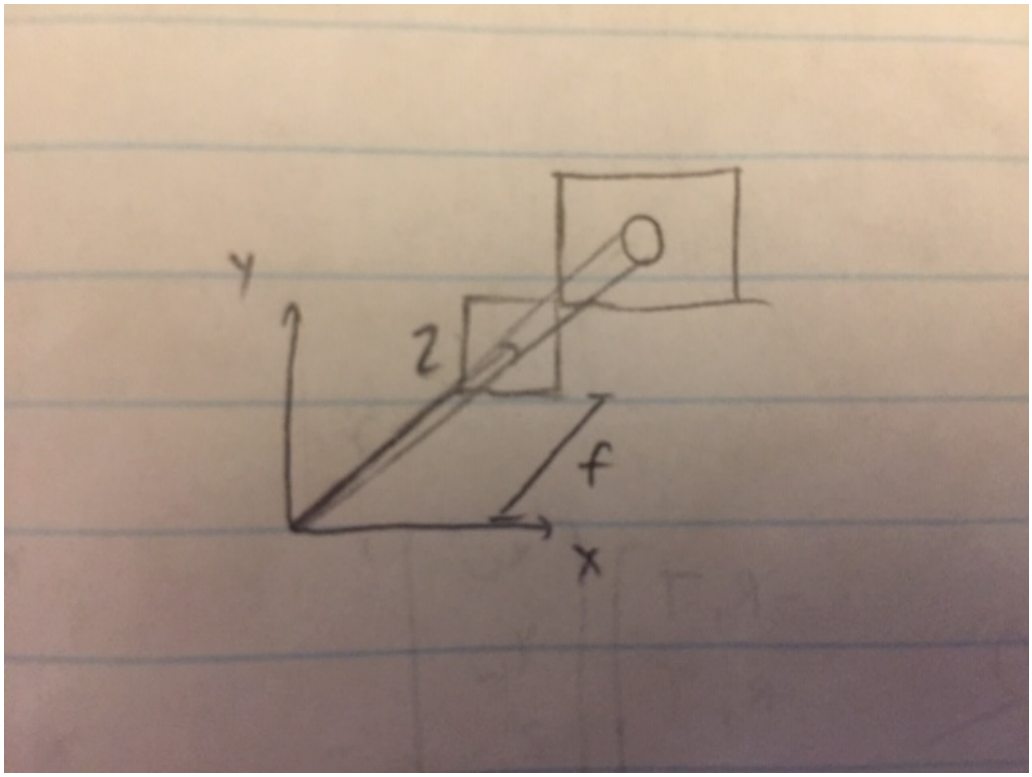

Figure 4: Projection onto image plane

4 Image to Pixel

To transform from image to pixel, we have to divide the image x coordinate by the width of a pixel and the image y coordinate by the height of a pixel. This will tell us how many pixels away a point in the image is from the center of the image. Let the width of a pixel by s_x and the height of a pixel by s_y. Also, note that we must negative the image y coordinate because the pixel u axis points down instead of up. Right now we have $u = -\frac{y}{s_y}, v = \frac{x}{s_x}$. However, we have to add $\frac{1080}{2} = 540$ to u and $\frac{1920}{2} = 960$ to v so that the pixel $(0, 0)$ is in the top left corner. The exact numbers come from the fact that each frame is 1080 by 1920 pixels.

$$u = -\frac{y}{s_y} + 540$$
$$v = \frac{x}{s_x} + 960$$

5 Extrinsic vs. Intrinsic Parameters

The rotation matrix R and the position of the camera $\boldsymbol{C}$ are considered **extrinsic parameters** in the sense that we set them. We can also obtain good estimates of what they are without doing any algebra or optimization. On the other hand, the focal length f and the pixel size s_x by s_y are **intrinsic parameters**, which means that they are properties of the camera. While the focal length is easily obtainable from the SOSUN website (7.36 mm unzoomed), the pixel size is unknown. However, we can estimate them using least squares!

We know the world coordinates of several points on the court, such as the free-throw rectangle and the white square on the backboard. Given our estimates for all of the other parameters, we can solve for the image coordinates (x, y) of these points relatively easily. We can also identify the pixel coordinates (u, v) of these points relatively accurately by examining a given frame (the camera is stationary). We have the equations:

$$v = \frac{x}{s_x} + 960$$
$$u = -\frac{y}{s_y} + 540$$

And we can write this as a matrix equation $\begin{pmatrix} v - 960 \\ u - 540 \end{pmatrix} = \begin{pmatrix} x & 0 \\ 0 & y \end{pmatrix} \begin{pmatrix} \frac{1}{s_x} \\ \frac{1}{s_y} \end{pmatrix}$. We have many points where we know (x, y) and (v, u), so we can make this a least-squares problem where we solve for

$\frac{1}{s_x}, \frac{1}{s_y}$!

$$\begin{pmatrix} v_1 - 960 \\ u_1 - 540 \\ v_2 - 960 \\ u_2 - 540 \\ \dots \end{pmatrix} = \begin{pmatrix} x_1 & 0 \\ 0 & y_1 \\ x_2 & 0 \\ 0 & y_2 \\ \dots & \dots \end{pmatrix} \begin{pmatrix} \frac{1}{s_x} \\ \frac{1}{s_y} \end{pmatrix}$$

Assuming we have estimated all of the other parameters correctly, we should be able to get accurate numbers for s_x, s_y, allowing us to transform any point on the court into pixels.

6 Alternative approaches

We might consider solving for *other* parameters using the least-squares method. However, I see two problems with this (feel free to let me know if these are valid concerns):

1. Since we have $x = \frac{f}{Z}X$, etc. it is not possible to solve for our extrinsic parameters using least-squares. Our pixel coordinates are *non-linear* functions of the elements in the rotation matrix R and translation vector $\boldsymbol{C}$.

2. Solving for these extrinsic parameters using least-squares might result in results that don't make physical sense. We might find that the residual is smallest with some rotation matrix that might be numerically valid but represents a physical situation in which the camera is not pointed anywhere near the hoop.

7 An example, using Camera 3

8 Software package

There is a package called OpenCV that I believe also exists in Python. It seems to have something to do with tracking objects using a camera. However, it might be too powerful and defeat the purpose of our project.

9 Resources

http://www.cse.psu.edu/~rtc12/CSE486/lecture12.pdf
http://www.cse.psu.edu/~rtc12/CSE486/lecture13.pdf
https://courses.cs.washington.edu/courses/cse455/09wi/Lects/lect5.pdf

https://www.cs.utah.edu/~srikumar/cv_spring2017_files/Lecture2.pdf
http://www.cs.toronto.edu/~jepson/csc420/notes/imageProjection.pdf
http://people.scs.carleton.ca/~c_shu/Courses/comp4900d/notes/camera_calibration.pdf
http://www.cs.ucf.edu/~mtappen/cap5415/lecs/lec19.pdf
https://www.cse.unr.edu/~bebis/CS791E/Notes/CameraParameters.pdf