Thesis Update

Carson Trego

December 10th, 2024

1 Relevant Objects

1.1 Choice

1.2 Choice

2 Pose Data

Each object of interest - be it an object belonging to one of the set of classes or the camera itself will be assigned a 6x1 vector:

$$\begin{bmatrix} p_x \\ p_y \\ p_z \\ p_{yaw} \\ p_{pitch} \\ p_{roll} \end{bmatrix}$$

Where: p_x is a distance from the origin in centimeters, with values ranging from positive infinity to negative infinity. In the context of this problem, a negative x value denotes that the object is behind the origin, and a positive x value denotes that an object is in front of the origin. p_y is a distance from the origin in centimeters, with values ranging from positive infinity to negative infinity. In the context of this problem, a negative y value denotes that the object is to the left of the origin, and a positive y value denotes that the object is to the right of the origin. p_z is a distance from the origin in centimeters, with values ranging from positive infinity to negative infinity. In the context of this problem, a negative z value denotes that the object is to the below the origin, and a positive y value denotes that the object is to the above the origin. p_{yaw} is the direction and proportion of a full rotation from the starting orientation, with values ranging from -0.5 to 0.5. In the context of this problem, increasing the yaw value would cause the camera to look right. p_{pitch} is the direction and proportion of a full rotation from the starting orientation, with values ranging from -0.5 to 0.5. In the context of this problem, increasing the pitch value would cause the camera to look up. p_{roll} is the direction and proportion of a full rotation from the starting orientation, with values ranging from -0.5 to 0.5. In the context of this problem, increasing the roll value would cause the camera to lean to the right.

The global position (x,y,z) and orientation (yaw, pitch, roll) are not estimable without some point of reference. This is similar to to the issue of estimability in linear models, in which linear combinations of parameters such as the intercept are not consistently estimable. The solution to the problem here is similar to the solution to estimating non-estimable terms in a linear model: simply

set a point to zero, even arbitrarily, and then communicate the results relative to other results. In this case, we need to set some point in the global space equal to zero, so we can estimate the relative change in position and orientation of two objects. For reasons outlined in later sections, we have decided to set the location and orientation of the camera equal to zero, such that the cameras vector is always the following:

$$\begin{bmatrix} p_x = 0 \\ p_y = 0 \\ p_z = 0 \\ p_{yaw} = 0 \\ p_{pitch} = 0 \\ p_{roll} = 0 \end{bmatrix}$$

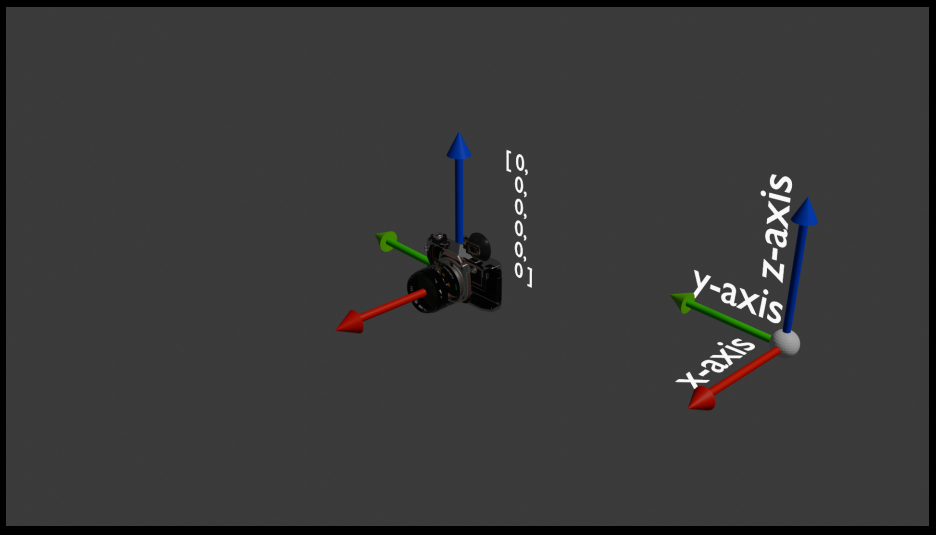

Figure 1:

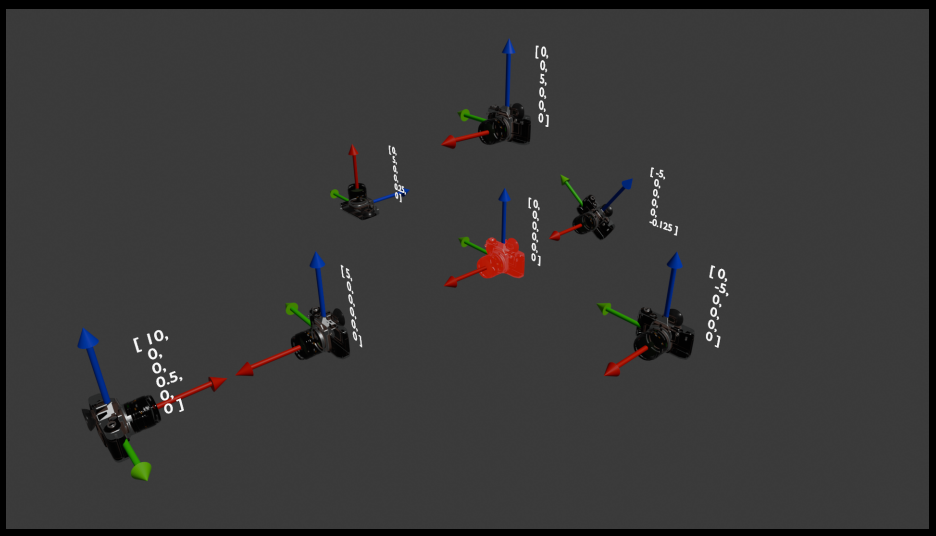

Figure 2:

3 The Origin

3.1 Choice

For this system, the origin will be defined on the camera, in which the x-axis will be extending out from the aperature of the camera, the y-axis will extend out to the right of the aperature of the camera, and the z-axis will be extending up from the viewing angle of the camera.

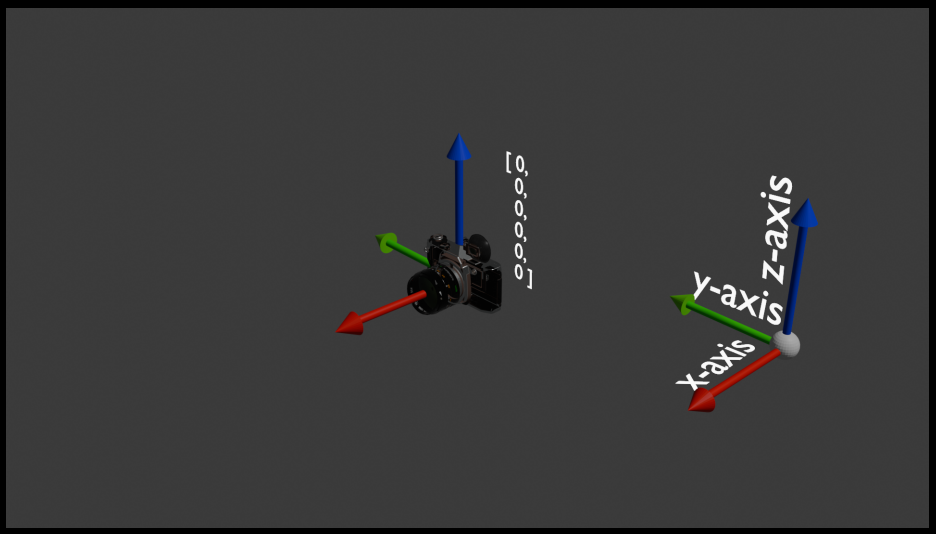

Figure 3:

3.2 Rationale

Centering on the camera, rather than another object or a fixed point in space, comes with a few advantages. Given that we are having a "relevant objects" system, as apposed to a system that directly involves the non-classified environment, allows the system to determine the relative location of objects without needing extra info about the location first. For example, if the origin was some fixed point in space, then the system would have to know where the camera is relative to that point in addition to determining where the object is. With the "relevant objects" system, all that the system needs to know is where each object is relative to the camera, and can operate in a completely unknown space.

The aforementioned advantages apply to a both systems where the origin is fixed on each object and systems where the origin is fixed on the camera. Here we chose to fix the origin on the camera to make processing multiple objects less complex.

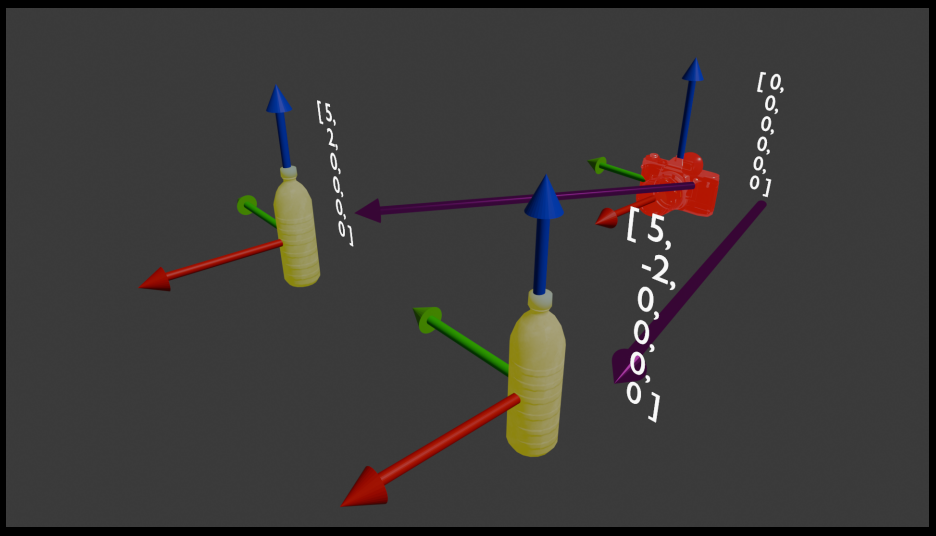

Figure 4:

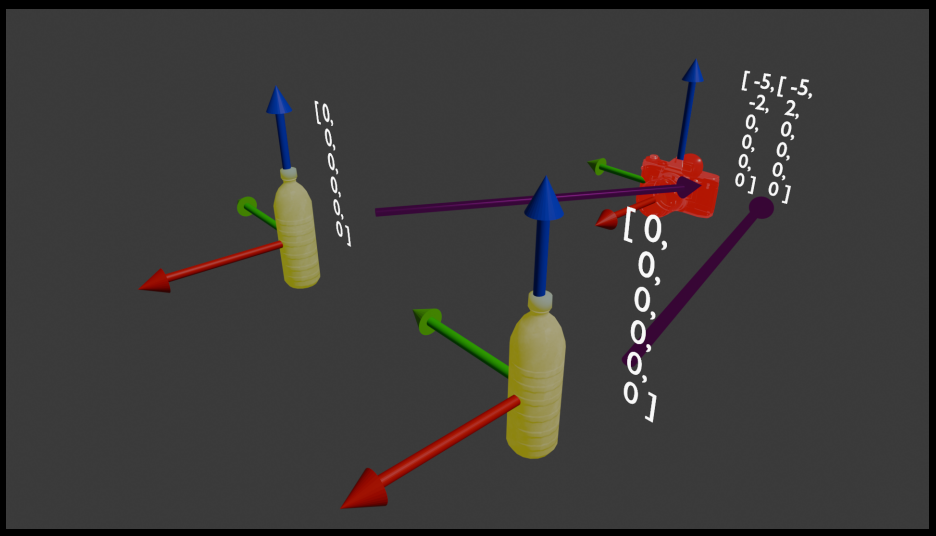

Figure 5:

4 Relative Pose

Relative pose will be defined by the the vector addition operation needed to change the camera vector, to the object vector. Each object will have a pair pose connected to the camera. To find the pose of one object, defined relative to a single camera, is given by the solution to this problem:

$$\begin{bmatrix} p_{camera\ x} \\ p_{camera\ y} \\ p_{camera\ z} \\ p_{camera\ yaw} \\ p_{camera\ pitch} \\ p_{camera\ roll} \end{bmatrix} + \begin{bmatrix} p_{pose\ x} \\ p_{pose\ y} \\ p_{pose\ z} \\ p_{pose\ yaw} \\ p_{pose\ pitch} \\ p_{pose\ roll} \end{bmatrix} = \begin{bmatrix} p_{object\ x} \\ p_{object\ y} \\ p_{object\ z} \\ p_{object\ yaw} \\ p_{object\ pitch} \\ p_{object\ roll} \end{bmatrix}$$

Given that we set the origin to be fixed with the camera:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} p_{pose\ x} \\ p_{pose\ y} \\ p_{pose\ z} \\ p_{pose\ yaw} \\ p_{pose\ pitch} \\ p_{pose\ roll} \end{bmatrix} = \begin{bmatrix} p_{object\ x} \\ p_{object\ y} \\ p_{object\ z} \\ p_{object\ yaw} \\ p_{object\ pitch} \\ p_{object\ roll} \end{bmatrix}$$

$$\begin{bmatrix} p_{pose\ x} \\ p_{pose\ y} \\ p_{pose\ z} \\ p_{pose\ yaw} \\ p_{pose\ pitch} \\ p_{pose\ roll} \end{bmatrix} = \begin{bmatrix} p_{object\ x} \\ p_{object\ y} \\ p_{object\ z} \\ p_{object\ yaw} \\ p_{object\ pitch} \\ p_{object\ roll} \end{bmatrix}$$

With the origin fixed on the camera, the pose becomes the transformation required to go from the cameras location and orientation, to the objects location and origin.

Looking back at the earlier figure showing the main camera and a set of camera objects, it can be seen that if the origin camera is moved to the location and orientation of one of the objects, the transformation needed is the pose of the object. The pose is framed as the displacement vector from the camera to the object.

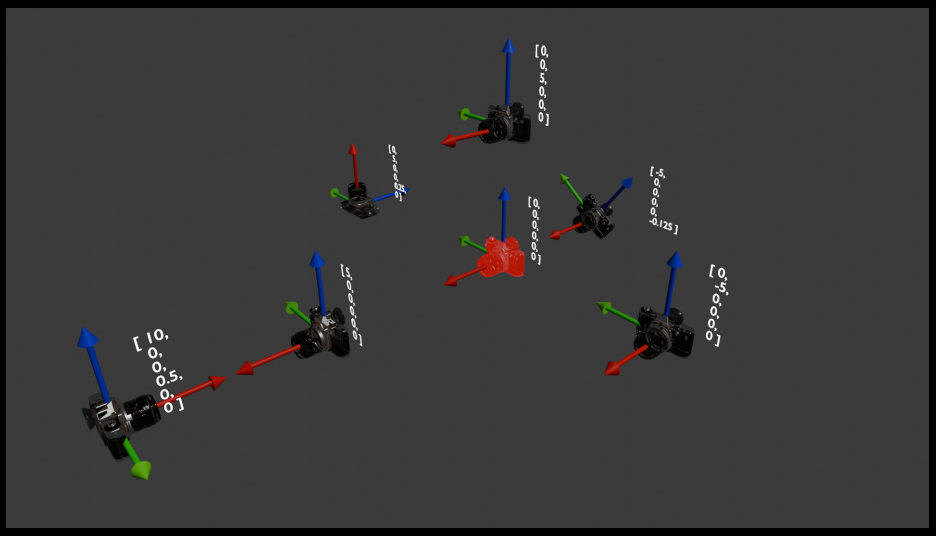

Figure 6:

5 Ongoing

There is a specific paper that I think is very interesting realted to this topic *PVNet: Pixel-wise Voting Network for 6DoF Pose Estimation* (Peng et al. 2018), and *PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes* (Xiang et al. 2018). These represent two very different methods, but within this project are similar in that they both use similar descriptions for pose. What is different about this method is the use of synthetic data, depth information, and simulating performance after the model is complete.

Depending on which one of these methods is employed, the transition from photo to pose could be very different. A hollistic method is closer to what we had first described, with a CNN looking at an image directly and extracting a pose, while a keypoint based method uses a much different method of determining pose from a CNN. What seems required in both cases is a vector field representation, segmentaton map, and discrete angle classes.)(This may be better presented than written, at least until I can make another animation)