A Geometric Approach to Human Pose Evaluation using Perspective Projection

Alex Liu

January 2026

Abstract

One of the most important aspects of computer vision being researched right now is human pose estimation. Most modern researchers aim to estimate 3D poses from 2D images, but this paper aims to do the opposite: projecting 3D joints onto a 2D plane. A geometric approach will be presented in this paper by first creating an orthonormal basis centered around the camera to represent the camera orientation. We will then map the 14 joint coordinates onto this new camera centered coordinate system for each of the 20 given frames. This approach will give us both the camera orientation, as well as a projected skeleton for each of the 20 poses that we will use to validate the process. The techniques explored will give insight into core pieces used in more complex computer vision techniques and 3D pose estimation algorithms.

1 Introduction

Human pose estimation is a primary point of research in the field of computer vision. It is highly valuable due to it's applications in a wide array of areas, ranging from healthcare and fitness to entertainment and human-computer interaction. In this paper, we aim to address a much simpler version of the task: given a fixed camera position as well as a set of 14 3D coordinates representing skeletal joints, find an appropriate camera orientation and a 2D projection of the poses onto the viewing plane of the camera.

To achieve this, we will first represent the original joint coordinates as a set of vectors. Then, we will construct 2 criteria to find an appropriate camera orientation. Finally, we will use perspective projection to map the 3D joints onto a 2D plane, accounting for both the depth of the subject as well as the focal length of the camera.

2 Methodology

2.1 Representing the original coordinates as vectors

The first step in our process will be to convert each of the original coordinates into position vectors in the vector space with basis $\{\hat{i}, \hat{j}, \hat{k}\}$. More formally, let

$$\text{joints} = \{\text{joint}_1, \text{joint}_2, \ldots, \text{joint}_{14}\}$$

be a set of 3D vectors, where

$$\text{joint}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}.$$

2.2 Determining the camera orientation

The next step to evaluate our skeletal pose is finding the camera orientation. There are a couple factors we need to consider to find an appropriate camera orientation, for which we can construct two criteria:

1. **Target centering:** The camera should point towards some "target" that captures the joints and
2. **Horizontally aligned:** The camera should be kept level with the floor.[1]

First, let's find an appropriate target. Many standard conventions used in other datasets such as Human3.6M protocol[1], use the pelvis joint as a "root" joint, which they take as the center of their coordinate system. This technique of fixing a global space relative to a root joint is also used in Martinez et al. [2] and their applications of predicting 3D joint locations from a 2D static image. As such, we will fix adopt a similar approach and pick the target of our camera to be at the hip joint.

ID	Joint Name	ID	Joint Name
0	Hip (Root)	7	Neck
1	Right Hip	8	Left Upper Arm
2	Right Knee	9	Left Elbow
3	Right Ankle	10	Left Wrist
4	Left Hip	11	Right Upper Arm
5	Left Knee	12	Right Elbow
6	Left Ankle	13	Right Wrist

Table 1: The joint IDs and associated names, as defined in the given dataset. Joint 0 (Hip) serves as the root for camera centering.

[1] Admittedly, this constraint is more of a soft constraint. However, it reduces an infinite number of solutions to a single, intuitive orientation that reflects the natural world most accurately.

Let us also define the **direction vector** to be

$$\hat{n} = \frac{\text{joint}_0 - c}{||\text{joint}_0 - c||} \tag{1}$$

where $c \in \mathbb{R}^3$ represents the position vector of our camera.

Now consider the orthonormal basis $\{\hat{u}, \hat{v}, \hat{n}\}$ representing the camera's orientation, where the subspace spanned by $\{\hat{u}, \hat{v}\}$ represents the viewing plane—the plane we want to project our 3D position vectors onto.

Since we want to also satisfy criteria 2, we want our basis to be "level" with the original coordinate system. That is, $\hat{u}$ should be orthogonal to the global z-axis. More formally,

$$\hat{u} = \frac{\hat{n} \times \hat{k}}{||\hat{n} \times \hat{k}||}, \tag{2}$$

where $\hat{k} = [0, 0, 1]^T$. More simply put, we fix $\hat{u}$ to be parallel to the original "ground plane"[2]. Finally, to calculate $\hat{v}$, we do a simple cross product:

$$\hat{v} = \hat{u} \times \hat{n} \tag{3}$$

to get all 3 vectors in our orthonormal basis.[3] Thus, for the camera orientation, we can construct the rotation matrix

$$R = \begin{bmatrix} \hat{u}_0 & \hat{u}_1 & \hat{u}_2 \\ \hat{v}_0 & \hat{v}_1 & \hat{v}_2 \\ \hat{n}_0 & \hat{n}_1 & \hat{n}_2 \end{bmatrix}. \tag{4}$$

2.3 Projecting onto the viewing plane

Now, we will apply a perspective projection for each of our original joint position vectors onto the viewing plane, or the subspace spanned by vectors $\hat{u}$ and $\hat{v}$. Let the **camera coordinate system** be the coordinate system formed from the orthonormal basis $\{\hat{u}, \hat{v}, \hat{n}\}$. It naturally follows that we can calculate the coordinates of each joint in our new camera coordinate system:

$$x_i' = (\text{joint}_i - c) \cdot \hat{u} \tag{5}$$

$$y_i' = (\text{joint}_i - c) \cdot \hat{v} \tag{6}$$

$$z_i' = (\text{joint}_i - c) \cdot \hat{n}. \tag{7}$$

Then, to account for the focal length of our camera, and assuming the original coordinates were given in millimeters, we get the following relation from similar triangles:

$$\frac{x_i''}{f} = \frac{x_i'}{z_i'}$$

[2]In the case that $\hat{k}$ is parallel with $\hat{n}$, the resulting $\hat{u}$ vector will be a zero vector. This simply means that our camera is pointing up/down, so $\hat{u}$ can be in any direction as long as its still orthogonal to our $\hat{n}$ vector. In code, we can simply check if the alignment is within some tolerance $\epsilon = 10^{-6}$, and assign $\hat{u}$ to some default vector $\hat{j} = [0, 1, 0]^T$.

[3]Note, we don't do $\hat{v} = \hat{n} \times \hat{u}$ because of the way the given coordinate system is oriented (in sort of a "left hand rule" way).

where x_i'' is the x coordinate after being perspective projected onto the viewing plane, and f is the focal length in millimeters. Rearranging, we get

$$x_i'' = f \cdot \frac{x_i'}{z_i'} \tag{8}$$

where x_i'' is the x coordinate of the original coordinate perspective projected onto the viewing plane[4]. Similarly, we also find that

$$y_i'' = f \cdot \frac{y_i'}{z_i'}, \tag{9}$$

giving us the final coordinates (x_i'', y_i''). In the implementation, rather than converting to pixel coordinates, we will simply use a python graphing library (like matplotlib) and scale the axes accordingly.

3 Results

3.1 Camera orientation and projected coordinates

Applying our methodology to the input dataset gave us the camera orientation for each of the frames. For example, Frame 10 gave us the resulting camera orientation matrix:

$$R = \begin{bmatrix} 0.90754762 & -0.41994919 & 0.00000000 \\ 0.14003629 & 0.30263091 & 0.94276422 \\ 0.39591307 & 0.85560342 & -0.33346009 \end{bmatrix}.$$

We also found the 2D coordinate projections for each joint for each frame. The following table shows a sample of the projected coordinates.

[4] Again, in the case that the "depth," or z_i' is 0, we set a lower bound on the value to some 10^{-6} to avoid division by zero.

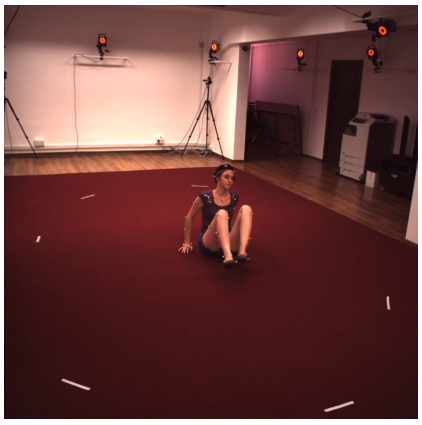
Joint Name	2D Coordinates
Hip	(0.000000000000000, -0.000000000000118)
Right Hip	(-33.200536334199143, 1.370140357789246)
Right Knee	(5.905634627286233, 66.974610388067333)
Right Ankle	(26.176238125299914, -54.881205791099639)
Left Hip	(32.631166708458707, -1.346641722501537)
Left Knee	(73.413420525409251, 81.974220296769502)
Left Ankle	(65.805206131269628, -36.601012709647094)
Neck	(1.744465022899954, 124.378619436825389)
Left Upper Arm	(33.598398396291870, 113.309422941174361)
Left Elbow	(29.868191335739404, 68.148710092265276)
Left Wrist	(36.216366214461388, 7.104451322489565)
Right Upper Arm	(-33.816294139860602, 111.418018181492840)
Right Elbow	(-70.196751059583917, 55.243424264894742)
Right Wrist	(-68.975593571265335, -8.674380071390580)

Table 2: The calculated 2D coordinates for each joint for Frame 10.

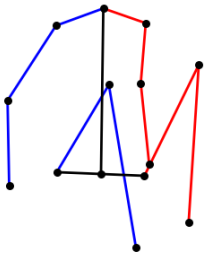
The full csv table containing the rotation matrix representing the camera orientation, as well as the projected 2D coordinates for each joint, for each frame, can be found here.

3.2 Visual projections

And finally, our methodology also gives us a comparison between the original image and the and skeletal projection plotted on a 2D graph. Figure 1 shows an example projection derived from a set of joints.



(a) Input Frame Pose



(b) Projected 2D Skeleton

Figure 1: Example projection for Frame 10. The resulting skeletal structure (b) aligns well with the pose of the original image (a).

The full list of side-by-side comparisons for each frame can be found here.

4 Discussion

The side-by-side comparison seems to give accurate results. No analysis (i.e. overlaying and error calculation) was done in this work, as it was deemed outside the scope of this project.
Surprisingly though, despite this project being a huge simplification of the more complicated task of true pose estimation, papers such as VideoPose3D [3] use an autoencoder model, where the decoder model is a projection layer that converts their constructed 3D coordinates back to 2D. This reflects real use cases of techniques discussed in this paper, specifically for semi-supervised learning models in the space of pose estimation.

5 Conclusion

In this paper, we successfully mapped 3D coordinates to 2D camera projections by calculating an orthonormal basis and then performing a perspective projection. Visual comparisons have confirmed reasonable levels of accuracy, with the discussed techniques having possible applications in more advanced computer vision techniques.

References

[1] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1325–1339, 2014.

[2] J. Martinez, R. Hossain, J. Romero, and J. J. Little, "A simple yet effective baseline for 3d human pose estimation," 2017.

[3] D. Pavllo, C. Feichtenhofer, D. Grangier, and M. Auli, "3d human pose estimation in video with temporal convolutions and semi-supervised training," 2019.