Beyond the Horizon

Camera Pose and 3D Scene Reconstruction from Single-View Geometry

Connor Malley

November 24, 2022

1 Introduction

When taking an image of a scene, the camera projection matrix P maps a set of 3-dimensional world points $(X, Y, Z, W)^T$ to a corresponding set of 2-dimensional image points $(x, y, w)^T$. However, the 3D information of the scene can be recovered with knowledge of the camera matrix P, or rather knowledge of the 2D position of the vanishing points in each of the (X, Y, Z) directions in the image. Some extra information about the planar surfaces in the image is needed as well, which is shown in this report. By recovering the 3D positions of just a few select keypoints, 3D models of objects in the scene can be roughly reconstructed, ignoring any intricate details.

2 Defining a Coordinate System

2.1 Computing the Vanishing Points

Before recovering the 3D information of the scene, an origin $O = (O_x, O_y, O_w)$ must be defined within the image plane. This origin is selected such that the (X, Y, Z) axes of the 3D scene are intuitive, such as the corner of a room. In this project, I assume that the XY plane comprises the ground plane, and that all objects lie on or above the ground plane.

Once the origin and axes are selected, the location of the vanishing points in the image are computed. For example, to find the Y vanishing point v_y, a line l_y parallel (in 3D space) to the Y axis (which I will denote as L_Y) is selected. The location of v_y is then given by the point of intersection of these two lines, which maps to a finite location of $v_y = L_Y \times l_y$ in the image. The same process is used for the other axes as well, giving v_x and v_z. The camera projection matrix is then given by $P = [\alpha v_x, \beta v_y, \gamma v_z, O]$, where (α, β, γ) are scale factors that must be computed.

2.2 Computing the Scale Factors

To compute the scale factors for each axis, either some reference lengths must be known, or the sides of some object in the scene (such as a cube) can be assumed to be unit length. For simplicity, I used reference lengths along each of the axes from the origin, but the measurements can be done off the axes as well. The scales factors are computed using a construction of a cross-ratio with the reference lengths, since the value of the cross ratio is invariant after projection using the camera matrix. I will give an example using the Z axis. First, the vanishing line for the Z axis is computed by getting the line through the vanishing points v_x and v_y, given by $V_z = v_x \times v_y$. Two endpoints of the reference length Z_{ref} are then selected, denoted as t and b for "top" and "bottom". The line for this reference length is then intersected with V_z to give the third point for the cross-ratio, $v_l = (t \times b) \times V_z$. The final point for the cross ratio is the Z vanishing point v_z. In 3D space, the cross ratio results in γZ_{ref}, due to the fact that the vanishing points are actually at infinity in 3D space. Since the cross-ratio is invariant, computing the same cross-ratio in the image plane, and dividing by the reference length will give the scale factor γ. In this project, I use the following cross ratio:

(a) Example 3D Coordinate System

(b) Example Parallel Lines

Figure 1: Defining Coordinate System and Computing Vanishing Points

$$\frac{||t-b|| * ||v_z - v_l||}{||t-v_l|| * ||v_z - b||} = \gamma Z_{ref} \tag{1}$$

Once the value of γ is found using the reference length, the length of any other line segment along the Z axis can be computed as well, by taking the same cross-ratio and dividing by γ. The other scale factors α and β are computed in a similar manner.[1]

2.3 Computing 3D Coordinates

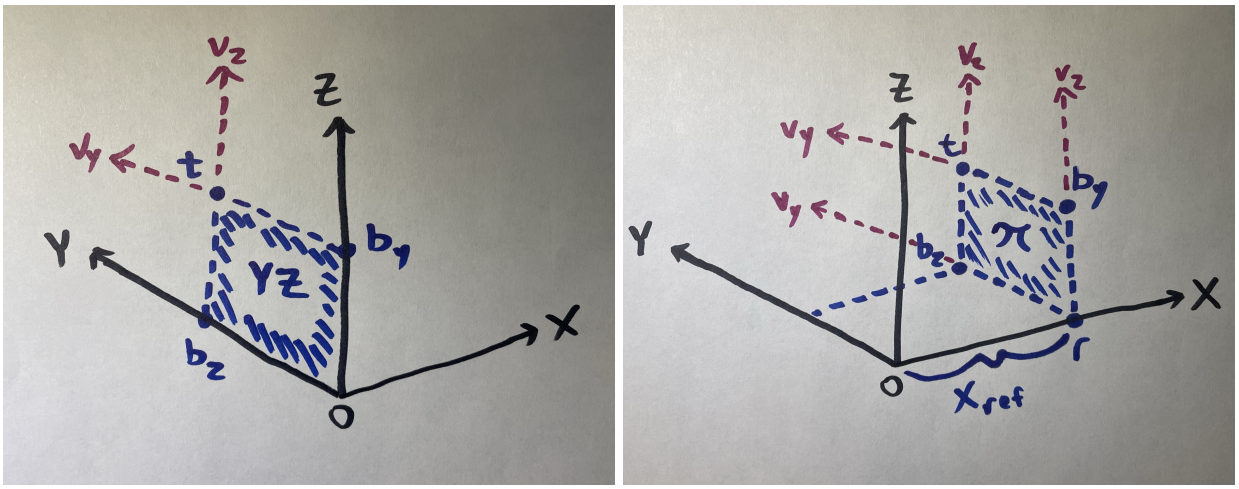
2.3.1 Coordinates in a Reference Plane

If computing the 3D coordinates of a point in one of the reference planes $(XY, XZ, or YZ)$, the method is relatively simple. For example, if computing the position of a point in they YZ plane, given by $p = (p_x, p_y, p_z, 1)^T$ we can start by assuming that $p_x = 0$. Assume the projected point is given in the image plane as $t = (x, y, 1)$. The line from t to v_y in the image plane is computed, and intersected it with the Z axis, given by $b_y = (t \times v_y) \times L_Z$. Similarly in the Y direction, we have that $b_z = (t \times v_z) \times L_Y$. This gives the points t and b for the cross-ratio in each direction. The lines between t and $\{b_y, b_z\}$ are intersected with the vanishing lines, giving the other points v_{ly} and v_{lz}. The coordinates p_y and p_z are then given by the following equation (after ensuring each point has a scale of 1):

$$p_x = 0, \qquad p_y = \frac{||t-b_y|| * ||v_y - v_{ly}||}{||t-v_{ly}|| * ||v_{ly} - b_y||}/\beta, \qquad p_z = \frac{||t-b_z|| * ||v_z - v_{lz}||}{||t-v_{lz}|| * ||v_{lz} - b_z||}/\gamma \tag{2}$$

2.3.2 Coordinates out of a Reference Plane

If the coordinates do not lie in one of the reference planes, but rather some other plane parallel to a reference plane, we must also know the distance of that plane from the reference plane. The distance of the planes could also be measured in the image using a cross-ratio, which simplifies things since the measurement doesn't have to be taken in the real world. In this case, there is a similar process. Again using the example relating to the $YZplane$, assume the point $p = (p_x, p_y, p_z, 1)^T$ lies on some plane π parallel to the YZ plane, and that π is a distance of X_{ref} from the YZ plane. In the image, the projected point is given as $t = (x, y, 1)$. Assume the plane π intersects the x-axis at the point $r = (r_x, r_y, 1)$ in the image. Then, lines can be drawn from r to the vanishing points v_y and v_z, call them R_y and R_z. Similarly, lines can be drawn from t to the vanishing points v_y and v_z, call them T_y and T_z. The lines from t and r are then intersected, where $b_y = T_z \times R_y = (t \times v_z) \times (r \times v_y)$ and $b_z = T_y \times R_z = (t \times v_y) \times (r \times v_z)$. This gives the point t and b that are needed to compute the cross-ratio in both the Y and Z directions, thus p_y and p_z are given by Equation 2, and $p_x = X_{ref}$.



(a) Measurement in YZ Plane

(b) Measurement in plane π parallel to YZ

Figure 2: In reference plane vs. out of reference plane measurements

2.3.3 Finding the Camera Centre

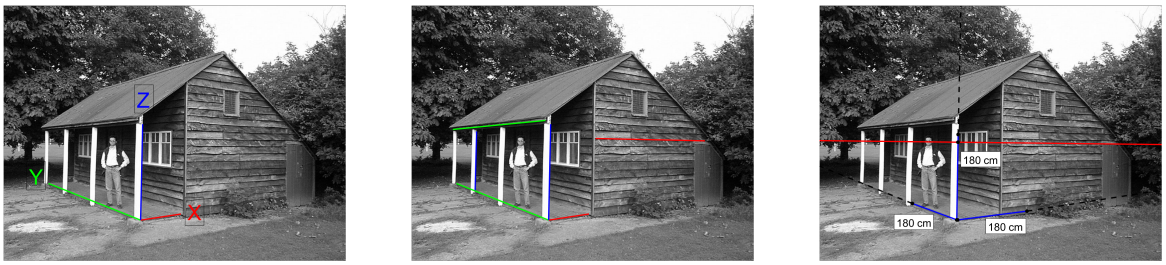
The camera matrix $P = [p_1, p_2, p_3, p_4]$ can be written in terms of the vanishing points, such that $P = [\alpha v_x, \beta v_y, \gamma v_z, O]$ where O is the image of the origin. The location of the camera centre is then given by the right null space of P. Once the scale factors α, β, γ are found, the camera centre can is recovered by constructing P and getting the right null-space of P using singular value decomposition. [2]

3 Methods

All of the 3D points were selected using one of the methods described in the previous section. Allowing points to be selected on one of the reference planes directly made computation a bit easier in some cases, since it did not require the specification of a reference length for the plane. A selection of keypoints were manually selected, and reference lengths between planes were measured using the same method of cross-ratios in the program. All points and line segments were then plotted using plot3 in MATLAB, and the planes were colored using fill3. In the process I made some assumptions, such that the back side of the shed had the same structure as the front side, and that the poles were evenly spaced and identical. In the 3D model of the television stand, I also assumed that the legs were identical, and I calculated their positions by simply measuring their distance in the program, and translating them along the XY plane. When selecting any person in a scene I also assumed that they were two-dimensional, which of course led to some slight errors.

4 Results

4.1 Shed Scene



(a) X, Y, and Z Axes of the coordinate system (b) Lines used for finding the vanishing points (c) Reference Measurements and XY plane vanishing line

Figure 3: Set-up of coordinate system for shed scene

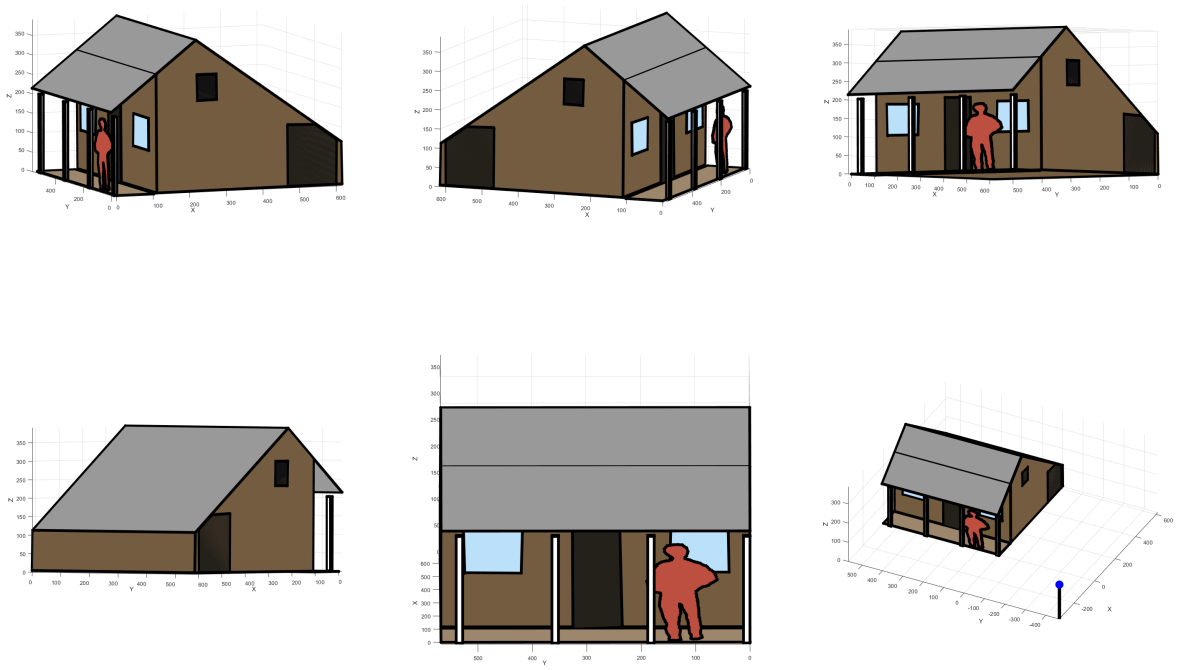


Figure 4: Views of shed 3D model (last image shows the camera centre)

This was one of the reference images using by Criminisi in [1]. The model looks fairly accurate, though the only scale I had to go off of was the height of the man (180 cm). I estimated the same distance in the other directions. Criminisi computed the camera centre at $\tilde{C} = (-381.0\ cm, -653.7\ cm, 162.8\ cm, 1)^T$ whereas my computed camera centre lies at $\tilde{C}' = (-344.3\ cm, -466.9\ cm, 179.7\ cm, 1)^T$. This is off by a bit, but it is to be expected since I only estimated the lengths in the X and Y directions.

4.2 TV Stand

(a) X, Y, and Z Axes of the coordinate system

(b) Lines used for finding the vanishing points

(c) Reference Measurements and XY plane vanishing line

Figure 5: Set-up of coordinate system for TV Stand scene

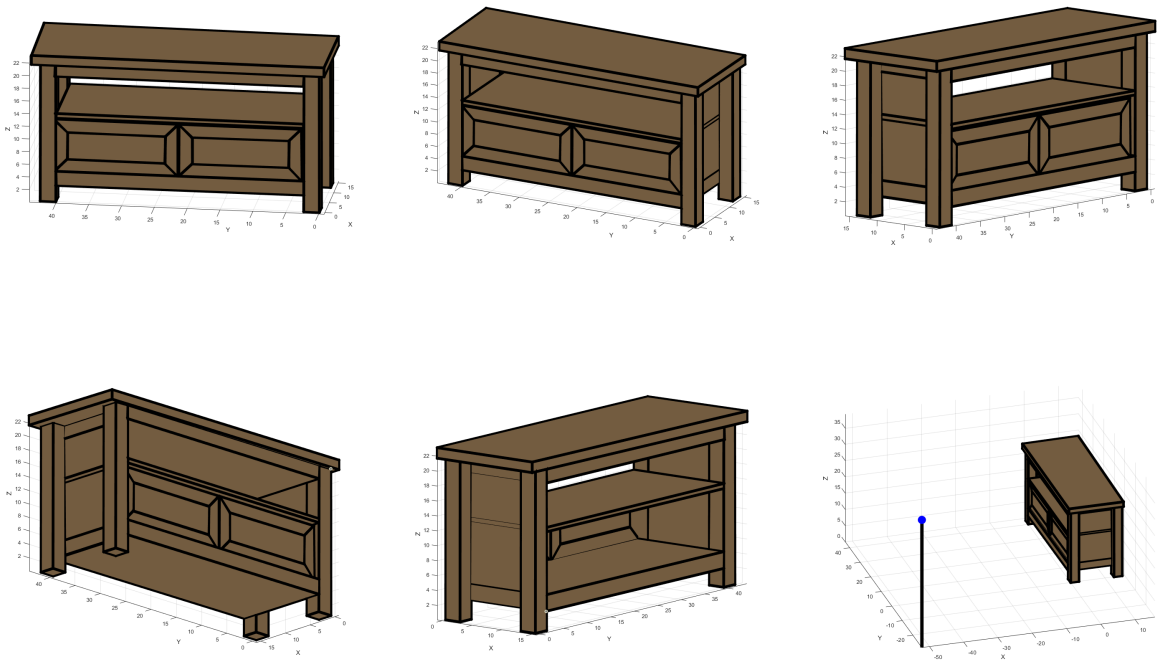


Figure 6: Views of TV Stand 3D model (last image shows the camera centre)

I took this image of a television stand. I chose this since it has a simple rectangular structure, but it still has some intricate details such as the grooves in the doors and the slightly larger top. I left the back off of the model to give a better view of the inside. I measured the camera centre to be $\tilde{C} = (-51.5\ in, -27.5\ in, 38.5\ in, 1)^T$ and the computed camera centre was $\tilde{C}' = (-52.2\ in, -28.4\ in, 39.3\ in, 1)^T$. This is within an inch in each direction, so I attributed the error to errors in the vanishing points due to me selecting the parallel lines manually.

4.3 Painting

(a) X, Y, and Z Axes of the coordinate system (b) Lines used for finding the vanishing points (c) Reference Measurements and XY/YZ plane vanishing lines

Figure 7: Set-up of coordinate system for Painting scene

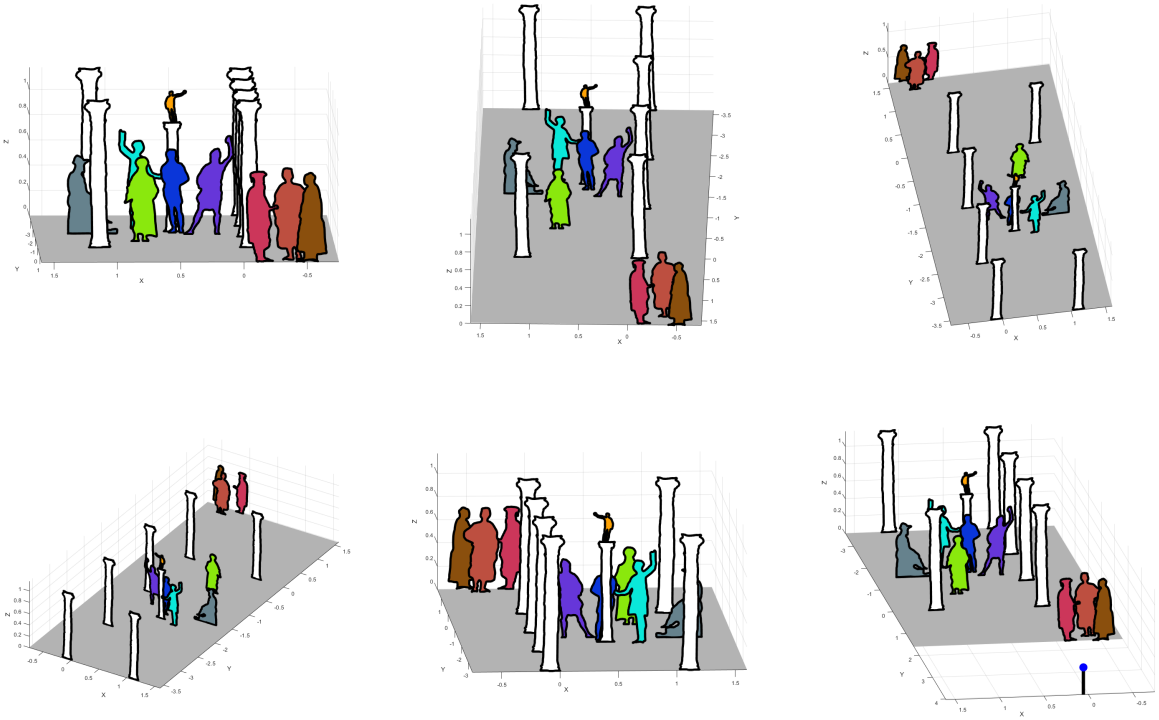


Figure 8: Views of painting 3D model (last image shows the camera centre)

This was another image used by Criminisi in [1], which is the painting "La Flagellazione di Cristo" by Piero della Francesca. Here I used one red square on the floor as the unit length, and simply estimated the same length in the Z direction, which is about the height of a pillar. Here, I computed the camera centre to be $\tilde{C}' = (0.056,\ 4.029,\ 0.313,\ 1)^T$. The reconstruction looks fairly accurate, though I ran into some issues with the sign of the X and Y positions, due to the nature of the vanishing points. Thus, I had to manually flip the signs some of the positions to put them in their correct place. Also, the model was extremely laggy in MATLAB due to the large number of points, so I ignored the detailed reconstruction of the floor. This was likely due to using fill3 rather than a direct texture mapping, though the 3D model still turned out fairly well. There is again some error in assuming that each person or pillar is 2D in structure.

5 Conclusion

Overall, I think the 3D modeling went well, especially when I had accurate reference measurements. The shed scene is accurate in the Z direction, but slightly distorted in the X and Y directions. The TV stand scene is the most accurate since I had exact reference measurements. Lastly, the painting scene is fairly accurate in all directions since I had a square on the XY plane to use as unit length. The main challenge was spending time to click the points manually, and making sure to do it precisely. I had many times where the model was severely distorted, and I had to restart the process. I made sure to save my progress whenever I made good measurements to prevent having to do this over and over. The main issues I ran into were due to the ordering of the cross ratio, and the positions of the vanishing points. Sometimes the vanishing points were in the positive directions, and sometimes they were in the negative direction. I ended up having to manually swap the signs of points from positive to negative in some cases. This is since the value of the cross ratio is always positive when using the L2 norm. If I had more time, I would implement a way to do this automatically based on the position of the chosen points related to the positions of the vanishing points. For some reason, I could also not get the correct measurements unless I changed the ordering of t and b in the cross-ratio. I would investigate this further if I had more time, as it was probably due to some bug in my program, but I was getting the correct measurements so I figured it was irrelevant for the purpose of the project. Finally, the method I used for filling in the polygons seemed to slow down the program quite a bit, so I would implement an actual texture mapping by first affinely rectifying each object and then cropping the texture. All of these implementation issues aside, this project went well, even if it was a bit tedious at times. My automated method of selecting points on some plane actually worked very well in practice, and this sped up the process this I did not have to manually input the "bottom" points in the cross-ratios. If you look closely at the 3D models, you can see that even some small details were captured, such as the golden statue in the painting, or the slight extension of the top of the TV stand.

References

[1] A. Criminisi. *International Journal of Computer Vision*, 40(2):123–148, 2000.

[2] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2004.