Introduction

This handout contains detailed explanations and derivations of the key concepts you need to understand in order to complete your final assignment in CS148. Much of the content was drawn from the articles and books in the References section. See Kanatani’s article [3] for an excellent tutorial on computational projective geometry, and Criminisi et al. [1] for more details on performing measurements using a single photograph. Finally, Hartley & Zisserman’s book [2] is an excellent reference on all things related to determining camera properties and reconstructing scene elements from one, two, or many images.

1 OpenGL Projection

Projection and View Matrices

Recall that vertices in OpenGL are first sent through a GL_MODELVIEW matrix and then a GL_PROJECTION matrix. When we discuss camera projection in this assignment, we include both the perspective projection and the transform due to the position and orientation of the camera. Thus, we consider the perspective projection matrix \( M_p \) and the view matrix \( M_v \) as follows:

\[
M_p = \begin{pmatrix}
\frac{f}{\alpha} & 0 & 0 & 0 \\
0 & f & 0 & 0 \\
0 & 0 & \gamma & \delta \\
0 & 0 & -1 & 0
\end{pmatrix}
\]

where \( \gamma = \frac{- (d_f + d_n)}{d_f - d_n} \) and \( \delta = \frac{-2d_fd_n}{d_f - d_n} \) \hspace{1cm} (1)

\[
M_v = \begin{pmatrix}
R & T \\
0^T & 1
\end{pmatrix}
\]

where \( T = -RC \) \hspace{1cm} (2)

From these matrices, we can see that we have three key unknowns to solve for: the camera’s focal length \( f \), the camera orientation \( R \), and the camera position (center of projection) in world coordinates \( C \). The dimensions of the photograph gives us \( \alpha \), the aspect ratio (width/height). The third row of \( M_p \) determines what gets stored in the depth buffer, and is not intrinsic to the camera projection itself. It contains the distance to the near and far clipping planes, \( d_n \) and \( d_f \) respectively, which we can set arbitrarily depending on the scale of the scene and models we wish to render.
Image and Clip Coordinates

The OpenGL matrices we are studying and will manipulate take world coordinate points to clip coordinates in the range $[-1,1]$. We want to set the OpenGL viewport to the exact size of the image, so that the projected point $(-1,-1)$ is on the bottom-left of the image, and the projected point $(1,1)$ is on the top-right. Thus, it is imperative that any points you select, define, or compute on the photograph be specified in these clip coordinates.

Perspective Projection

A geometric interpretation of perspective projection through a pinhole camera is shown in the diagram below. The perspective projection matrix takes a point $(X,Y,Z)$ in 3D camera (or eye) coordinates and maps it to a point $(x,y)$ on the image plane. In camera coordinates, the image plane is parallel to the XY-plane, and sits at a distance $f$ (focal length) away from the origin in the -Z direction. As shown, the top of the projected image has a 3D coordinate of $(0,1,-f)$ and the right has a 3D coordinate of $(\alpha,0,-f)$. The aspect ratio $\alpha$ appears because we are projecting a rectangular region of the image plane into the unit clip coordinates of range $[-1,1]$ on both axes.
2 Projection Recovery

Assuming that image pixels are square and that the optical axis passes through the center of the image, we can compute the camera projection matrix from just a few pieces of information. We need to know the projected positions of two orthogonal vanishing points in the photograph, which then become the X and Y directions in the world coordinate system. We also choose an arbitrary point on a reference plane (usually the ground) from the image that corresponds to the projection of the world coordinate origin, which becomes our reference point for placing objects.

Vanishing Points and the Focal Length

We can think of a point at infinity as a point where parallel lines meet. In 3D space, the point can be represented using homogeneous coordinates as \((X, Y, Z, 0)\). The vector \(V = (X, Y, Z)\) is the direction of parallel lines that intersect at the point at infinity. Because \(W = 0\), the projection of points at infinity are not affected by the position of the camera. Like stars in the night sky, they can be used for directional reference. Parallel lines projected with perspective often do meet at finite points, and we call these \textit{vanishing points}.

Suppose we can find the positions of two orthogonal vanishing points on the photograph: \(v_1 = (x_1, y_1)\) and \(v_2 = (x_2, y_2)\). Then their positions on the image plane in 3D camera coordinates are, respectively, \(V_1 = (\alpha x_1, y_1, -f)\) and \(V_2 = (\alpha x_2, y_2, -f)\).

Because the vanishing points are orthogonal, the dot product of the directions to these points in 3D must be zero. We can use this to solve for the focal length \(f\) as follows:

\[
V_1 \cdot V_2 = 0 \\
\alpha^2 x_1 x_2 + y_1 y_2 + f^2 = 0 \\
f = \sqrt{-\alpha^2 x_1 x_2 - y_1 y_2}
\]
Camera Orientation

Our next task is to recover the matrix $R$ which determines the camera orientation. We can write $R$ in terms of its three orthonormal basis vectors:

$$R = (r_1 \ r_2 \ r_3).$$  \hspace{1cm} (6)

If we let $v_1$ and $v_2$ be the vanishing points for the world $X$ and $Y$ directions respectively, then $R$ must rotate the direction $(1, 0, 0)$ to line up with $V_1$ and $(0, 1, 0)$ to line up with $V_2$. Therefore,

$$r_1 = \frac{V_1}{\|V_1\|} \text{ and } r_2 = \frac{V_2}{\|V_2\|}. \hspace{1cm} (7)$$

The columns of a rotation matrix are orthonormal, so we set $r_3 = r_1 \times r_2$ to complete $R$.

Camera Position

The last unknown we want to solve for is the camera position. Since the camera position is relative to the 3D positions of the objects in the scene, we can determine the camera center by choosing where on the photograph the world origin will be projected to. Suppose the world origin $O = (0, 0, 0)$ gets projected to the point $o = (x_o, y_o)$ in the photograph. In camera space, this point has a 3D coordinate on the image plane of $O' = (\alpha x_o, y_o, -f)$, so $T$, the translation component of the view matrix $M$, must move the world origin to a point on the ray $\lambda O'$. Hence,

$$T = -RC = \lambda \begin{pmatrix} \alpha x_o \\ y_o \\ -f \end{pmatrix}. \hspace{1cm} (8)$$

There is one remaining problem though: we still do not know the scale factor $\lambda$. If you think about it, you may realize that we can only recover the camera geometry up to an unknown scale with the information we have so far, because we know nothing about the physical size of the objects in the scene. You can arrive at the same picture by increasing the size of all the objects in the scene at the same time that you move the camera back along its optical axis.

To recover the projection matrix correct to scale, we can use one last piece of information: the height above the reference plane from which the photograph was taken. Then we can write $C = (X_c, Y_c, Z_c)$, where $Z_c$ is known, and solve for the scale:

$$-RC = \lambda O'$$

$$-C = \lambda R^T O' \hspace{1cm} (9)$$

$$\lambda = -Z_c / r_3 \cdot O' \hspace{1cm} (10)$$

If the height of the camera is unknown, but you know another reference distance in the scene, you can still solve for the scale factor. The technique in the next section can be applied if you know the height of another reference point from the ground.
3 Finding 3D Positions

At this point, you have both the projection and view matrices, and if you drop those into your OpenGL matrix stacks, any geometry you render should have a perspective that matches the photograph. Our goal is to render virtual objects onto various flat surfaces at different heights in the photograph, but where in 3D do we put our models? Rather than placing objects using a trial-and-error process by which you guess a 3D position, render the scene to see what it looks like, then move it bit, we will develop some machinery that allows you to place objects by picking points on the photograph itself.

Plane Heights

We are interested in placing objects onto flat surfaces in the scene that are parallel to the reference plane, so the first thing we want to do is to measure the heights of these surfaces. Assuming that the reference XY-plane is horizontal, we can determine the height of another parallel plane if we can see a point on the reference plane and a point on the second plane that is directly above or below it.

While perspective projection preserves neither angles nor distances, it does preserve a quantity known as the cross-ratio of four points on a line. We can use an invariant cross-ratio of four collinear points on a vertical line in 3D and the same corresponding points on the projected image to measure a vertical height in the scene. The cross-ratio is of a point $x$ on the reference plane, a point $x'$ on the second plane directly above $x$, the intersection $c$ between the vertical line and the vanishing line of the horizon, and the vanishing point of the upward direction $v_3$, as shown in the diagram below. Their corresponding points in 3D are denoted $X$, $X'$, $C$, and $V_3$.

From the photographs, you can select the points $x$ and $x'$. The horizon $\ell$ is the line that passes through the first two vanishing points $v_1$ and $v_2$. Take $c$ as the intersection between $\ell$ and the line that passes through $x$ and $x'$. Because vertical lines on the photographs are often near-parallel,
it is better to derive the last vanishing point \( v_3 \) from quantities we know, rather than to attempt to measure it. We know the point at infinity in the vertical direction \( V_3 = (0,0,1,0) \), and by projecting through the matrices we recovered, we obtain, in 2D homogeneous coordinates:

\[
\bar{v}_3 = \begin{pmatrix}
\frac{f}{\alpha} & 0 & 0 & 0 \\
0 & f & 0 & 0 \\
0 & 0 & -1 & 0
\end{pmatrix} r_3 \\
= \begin{pmatrix}
fr_{13} \\
fr_{23} \\
-1
\end{pmatrix}^T, \\
\text{so } v_3 = \begin{pmatrix}
-fr_{13} \\
-fr_{23} \\
-1
\end{pmatrix}^T. \\
(12)
\]

In world coordinates, \( X \) is on the reference plane, so its height is 0. \( X' \) is on the second plane at an unknown height of \( Z \) which we wish to measure. Because \( C \) projects to a point on the horizontal vanishing line, we know it is at the same height as the camera from the ground, or \( Z_c \).

The final cross-ratio is written as

\[
\frac{d(x, c)d(x', v_3)}{d(x', c)d(x, v_3)} = \frac{d(X, C)d(X', V_3)}{d(X', C)d(X, V_3)}, \\
(15)
\]

where the function \( d(a, b) \) evaluates the distance between points \( a \) and \( b \). Because \( V_3 \) is a point at infinity, we know the ratio \( \frac{d(X', V_3)}{d(X, V_3)} = 1 \). By the points chosen, we also know that \( d(X, C) = Z_c \) and \( d(X', C) = Z_c - Z \). Substituting these quantities, we can solve for the height of \( X' \) as

\[
\frac{d(x, c)d(x', v_3)}{d(x', c)d(x, v_3)} = \frac{Z_c}{Z_c - Z} \\
Z = Z_c \left(1 - \frac{d(x', c)d(x, v_3)}{d(x, c)d(x', v_3)}\right) \\
(16)
\]

Note that the exact ordering of the four points \( x, x', c, \) and \( v_3 \) on the line in your photograph may vary, depending on where the planes and vanishing point are located. If this is the case, you may have to adjust the terms in equation (15) to reflect the ordering. Be sure to adjust the terms on both sides of the equation, and note that isolating for \( Z \) in your new equation may yield a result that is slightly different from (17).

Observe that this cross-ratio can also be used to solve for the camera height if it is unknown, but the height of another reference point is known instead. Once the camera height is computed, it can in turn be used to solve for the height of other points in the scene.

**Image to World Position**

A point on the projected image plane corresponds to a ray in 3D space, so in general it is not possible to determine a unique 3D point that corresponds to a projected point in our photograph. However, if we know the height at which the point sits in 3D, then we can intersect the ray with the horizontal plane at that height to obtain the corresponding position in space.

Suppose you pick a point \( p = (x_p, y_p) \) from the photograph, and you know that it is located at a height of \( Z_p \) from the world reference plane. In camera coordinates, we know the corresponding
3D point projects to the point $P_c = (αx_p, y_p, -f)$ on the image plane. Thus, the point must lie somewhere on the ray $tP_c$ for some $t > 0$. If we use the inverse view matrix to rotate and translate the ray back into the world coordinate system, we get

$$P = C + tR^{-1} \begin{pmatrix} αx_p \\ y_p \\ -f \end{pmatrix}.$$  \hspace{1cm} (18)

Since you know $Z_p$ (and also $f$, $R$, and $C$ from the projection recovery step), you can use that to solve for the parameter $t$. Then substitute $t$ back into equation (18) to get $P$ in world coordinates.

4 Estimating the Light Direction

We will assume that the scene is illuminated by a single, directional light source. This assumption works well for photographs taken outdoors under direct sunlight. If we can find a point (say a corner) on a raised surface and the corresponding point on the reference plane where it casts a shadow, then we can compute the direction from which the light source is illuminating the scene. For example, if you can find a point $q$ on an object and a corresponding point $q_s$ on its shadow, you can use the method described in the previous section to compute the world coordinates of $Q$ and $Q_s$. The light direction is then simply $Q - Q_s$.

If instead you want to estimate the position of a point light source, then you can find two point-shadow correspondences, which give you two different rays in space. The light source position is the intersection of these rays (or the nearest point in between if they do not intersect exactly).

References

