MATH 136 Linear Algebra Camera Project

Josh Lando, Leo Carlin, Skye Luebbe Davidson

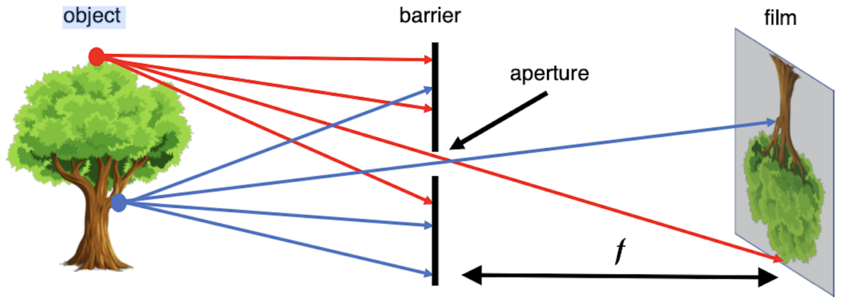
May 2025

Background and Significance

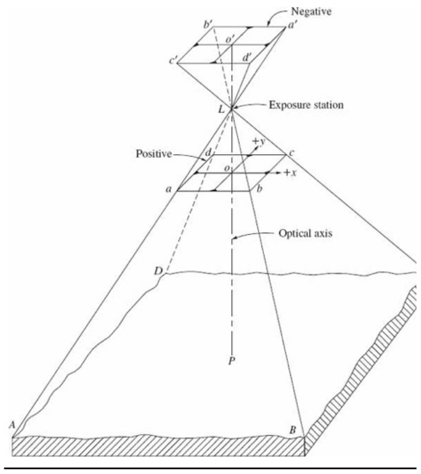
The Husky Satellite Lab (HSL) - which has current and former Math 13X students - is hoping to make its contribution to space exploration by developing an autonomous position and attitude determination system capable of navigating a satellite around the moon. HSL's work to build this system involves removing any dependency on external earth-based navigation infrastructure (for example, GPS or earthbound tracking stations). To do this, HSL is developing algorithms that can use images taken by cameras mounted on a satellite and calculate the position and attitude of the satellite. This project aims to explore the math behind cameras and how it can be used to solve this problem of figuring out a satellite's position and attitude.

Introduction to Cameras

One of the most simple models for understanding a camera is the **pinhole camera model**, named for having an infinitesimally small "pinhole" as the only gap for light to pass through. Pinhole cameras exist as purely theoretical models, but they provide a basis for working out the math governing real-life, lens-based cameras.



(a) Light from a point on the object being mapped to the film



(b) Diagram showing relationship between points in the negative and positive. **L** is the optical center.

Figure 1: The pinhole camera model

Figure 1a depicts a tree being imaged by a pinhole camera. In this system, 3D points from the scene emit light in every direction. As light travels toward the camera most of it is blocked by the barrier, theoretically allowing only one light ray from each point to pass through. This gap is called the **aperture**.

Additionally, all light rays entering the pinhole camera pass through one point, which is called the **optical center** of the camera. For pinhole cameras, the pinhole, aperture, and optical center are the same point. For lens-based cameras, the optical center and aperture may be different.

After passing through the optical center, the light then hits the **film or image plane** and is recorded as a 2D point. The actual image plane is behind the optical center of the camera. However, as Figures 1a and 1b show, the object appears upside down on an image plane behind the optical center. For this reason, the image plane is commonly referred to as the **negative**. As a common simplification, the image plane is drawn between the scene and the optical center of the camera, becoming the **positive or virtual image**.

Running through the camera, orthogonal to the image plane and going through the optical center is the **optical axis**. Where the optical axis meets the image plane is called the **principal point**, and is marked as o' in Figure 1b. **Focal length** describes the distance between the optical center of the camera and the image plane. In Figure 1a focal length is denoted by *f*, and in Figure 1b it is $||L-o||$. Throughout this project, we will use f to denote the focal length.

One way to think about how cameras work is that if we draw a straight line from a point on the image plane through the optical center out into the scene, that point will represent the first object the line hits. Following this thinking, we can think of cameras as a mapping between 3D points in the scene and 2D points on the image plane.

$$C : \mathbb{R}^3 \mapsto \mathbb{R}^2 \tag{1}$$

Note: $nullity(C) \geq 1$. This means we will lose information about the 3D scene.

Culture: This model has so far treated points, as, well, points. However, when taking images in real life, it is actually impossible to measure the individual RGB (color) values for a single point. To achieve a color image light filters are placed in the front of the image in a **bayer pattern** and software interpolates missing values in a process called mosaicing.

Introduction to Cameras as Linear Systems

Using the pinhole camera model, we can start describing the mapping from a 3D scene to a 2D pixel space with math. For this mapping we will ignore color and only measure the combined amplitude of light in the visible spectrum for each point (a black and white image).

Coordinate Systems

To explore this mapping, we first need to get a sense of the coordinate systems used. These coordinate systems account for the camera's view of objects, the mapping from Euclidean 3-space to Euclidean 2-space, and pixelation of the image.

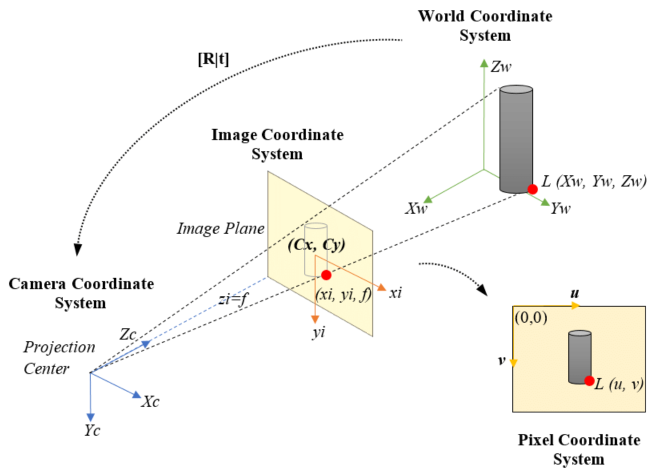


Figure 2: Camera coordinate systems

First, the **Camera Coordinates**. When thinking about the mapping between points in the scene to the virtual image it is convenient to have a coordinate system defined relative to the camera. As seen in Figure 2, the camera coordinate system has its origin at the optical center of the camera, z-axis along the optical axis, and x and y parallel to the virtual image.

Next, the **Film Coordinates**. This coordinate system captures the mapping of the 3D points onto a 2D plane located at $z = f$ in camera coordinates. This mapping preserves continuity and captures the effects of camera characteristics.

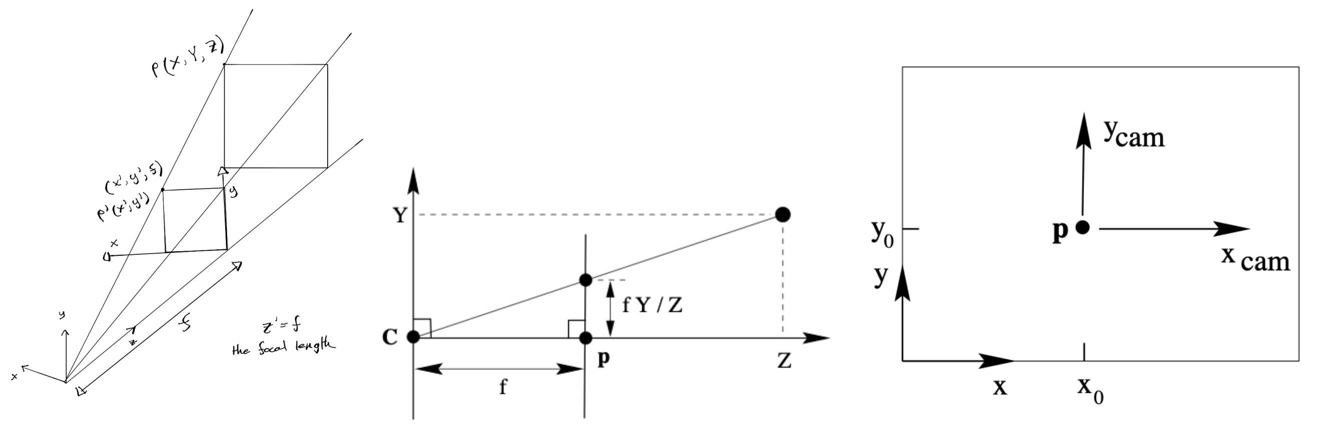
Pixel Coordinates mostly pertain to the pixelation of an image as well as how they are stored on a computer. The transformation from film coordinates to pixel coordinates are commonly combined with the transformation from camera to film coordinates, which is what we will do.

Finally, the **World Coordinates**. In general, points in the world are expressed in terms of a different coordinate frame than camera coordinates. The transformation from world coordinates to camera coordinates can be described as a translation and rotation as seen in Figure 2.

Transformations between coordinate systems

Camera Coordinates $\mapsto$ Film Coordinates

This transformation is responsible for mapping 3D points onto the 2D film plane. We start with the camera coordinates of a point $(X, Y, Z)^\top$ and the map onto the virtual image located at $z = f$. Using the law of similar triangles shown in Figures 3(a) and 3(b), we find that $(X, Y, Z)^\top$ is mapped to $(Xf, Yf, Z)^\top$ or in film coordinates, $P' = (x', y')^\top = (\frac{Xf}{Z}, \frac{Yf}{Z})^\top$.



(a) **P** is in camera coordinates, **P'** is in film coordinates

(b) **C** is the camera center and origin

(c) Pixel coordinates (x,y), film coordinates (x_{cam}, y_{cam})

Figure 3: Pinhole camera model geometry

Film Coordinates $\mapsto$ Pixel Coordinates

Figure 3c shows how the origin in camera coordinates is usually shifted from the optical center to the bottom left in pixel coordinates. Thus, the points are offset by a translation vector $[c_x, c_y]^\top$.

$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} f\frac{X}{Z} + c_x \\ f\frac{Y}{Z} + c_y \end{bmatrix} \tag{2}$$

Currently we are assuming that images have equal scales in both axial directions. However in modern digital cameras it is common to have non square pixels. To accommodate for this fact we introduce two new variables m_x, m_y with units $\frac{pixels}{distance}$. The transformation is now.

$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} fm_x\frac{X}{Z} + m_x c_x \\ fm_y\frac{Y}{Z} + m_y c_y \end{bmatrix} = \begin{bmatrix} \alpha_x\frac{X}{Z} + x_0 \\ \alpha_y\frac{Y}{Z} + y_o \end{bmatrix} \tag{3}$$

Now that we have defined a transformation from Camera Coordinates to Pixel Coordinates $P \mapsto P'$ we would like to find a convenient way to express this transformation. Unfortunately, our mapping divides one of the inputs, (Z), so it is not linear. However, we can use homogenous coordinates to make the transformation linear.

To go from Euclidean coordinates to homogeneous coordinates we introduce a new coordinate such that $P' = [x', y']^\top$ becomes $P'_h = [x', y', 1]^\top$ and $P = [X, Y, Z]^\top$ becomes $P_h = [X, Y, Z, 1]^\top$. As described above, $(v_1, v_2, ..., v_n)$ is equivalent to $(v_1, v_2, ..., v_n, 1)$. This equivalence only holds for one. Thus, from the arbitrary homogeneous coordinates $(v_1, v_2, ..., v_n, w)$ we get the Euclidean coordinates $(\frac{v_1}{w}, \frac{v_2}{w}, ..., \frac{v_n}{w})$. For more details on homogeneous coordinates, see Appendix A.

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} \alpha x' + Z x_0 \\ \beta y' + Z y_0 \\ Z \end{bmatrix} = \begin{bmatrix} \alpha & & x_0 & 0 \\ & \beta & y_0 & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4}$$

Now, writing

$$K = \begin{bmatrix} \alpha_x & & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix} \tag{5}$$

(3) now has the concise form

$$P' = K[I \mid 0]P = M_{\text{int}}P \tag{6}$$

The matrix K is commonly referred to as the **camera projection matrix**.

Complete Camera Matrix

Currently, the camera matrix contains 3 key pieces of information about the camera: principal point, focal length, and pixel size. This leaves out a key parameter that describe lens distortion- skew. This parameter exists due to the physics of lens-based cameras. Deriving the new camera matrix taking into account skew we get,

$$K = \begin{bmatrix} \alpha_x & s & c_x \\ 0 & \alpha_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

This complete camera matrix allows us to interpret finite projective camera matrices, but the explanation for how skew is dervied is beyond the scope of this paper.

World Coordinates $\mapsto$ Camera Coordinates

The transformation between the World Coordinates and the Camera Coordinates requires translation and rotation to align with the camera's optical center. Let W be the coordinates of a point in the World Coordinates and C_w be the World Coordinates of the optical center of the camera.

$$W = \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix}, \; C = \begin{bmatrix} (cx)_w \\ (cy)_w \\ (cz)_w \end{bmatrix} \tag{8}$$

Using vector addition, we can get close to C, the coordinate of the point in relation to the optical center of the camera

$$C \approx W - C_w \tag{9}$$

However, this only deals with translating the point between the two coordinate systems and does not account for the alignment of the Camera Coordinates themselves. So, we need to introduce a 3×3 rotation matrix R, to fully align the Camera and World Coordinates. This rotation may require rotating around all three axes.

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = R\left(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} - \begin{bmatrix} (cx)_w \\ (cy)_w \\ (cz)_w \end{bmatrix} \right). \tag{10}$$

However, due to the translation, this transformation is not linear. To fix this, we will again shift to a homogeneous coordinate system. We can now define the transformation from World Coordinates to Camera Coordinates as

$$\begin{bmatrix} C \\ 1 \end{bmatrix} = \begin{bmatrix} R & -RC_w \\ 0 & 1 \end{bmatrix} \begin{bmatrix} W \\ 1 \end{bmatrix} \tag{11}$$

Combining this with (5) leads to the formula

$$x = KR[I\,|\,-C_w]X \tag{12}$$

where x is the point in homogeneous pixel coordinates and homogeneous X is the point in world coordinates. A camera for which the K is of form (7) is called a finite projective camera model $P = KR[I\,|\,-C_w]$ and has 11 degrees of freedom: 5 for K (the elements f, c_x, c_y, m_x, m_y), 3 for R (the angles of rotation $\theta_x, \theta_y, \theta_z$), and 3 for C. The camera parameters contained within K are called the **internal camera orientation**. The parameters contained in R and C describe the **external camera orientation**. The left hand 3×3 sub matrix of P, KR, is non-singular (KR has a non-zero determinant). This follows from the fact that K is upper triangular and R is a rotation matrix. Any 3×4 matrix with a non singular 3×3 left sub matrix is in fact a **finite projective camera** model. This follows from the fact that a 3×3 non-singular matrix can be decomposed into an upper triangular and an orthonormal rotation matrix through a process called RQ decomposition (see Appendix C). The matrix P can therefore be written $P = M[I\,|\,M^{-1}p^4] = KR[I\,|\,-C_w]$ where p_4 is the last column of P. To summarize the set of finite projective camera matrices is identical to the set of homogeneous 3×4 matrices where the left hand 3×3 sub matrix is non-singular.

Focal Length Numerical Example

Let's examine how one of these parameters, the focal length f, affects the image taken by a camera. Let a simple camera have a optical center at

$$C = \begin{bmatrix} 0 \\ 0 \\ -3 \end{bmatrix}$$

in world coordinates. Note that, as with our previous math, Z is a horizontal axis.

Let the camera also have a focal length, f, of 4 units in world coordinates. Additionally, give the camera the principal point $(c_x,\ c_y) = (0,\ 0)$ - this means optical axis will be in the center of the image and a pixels per unit of world coordinate conversion of 400 pixels to one world unit ($m_x = m_y = 400$). This constructs the camera matrix

$$P = \begin{bmatrix} 1600 & 0 & 0 & 0 \\ 0 & 1600 & 0 & 0 \\ 0 & 0 & 1 & 3 \end{bmatrix} \tag{13}$$

Note that this example is aligning the world and camera coordinate systems to disregard rotation,
Consider a point given with homogenous world coordinates by

$$W_1 = \begin{bmatrix} 1 \\ 1 \\ 5 \\ 1 \end{bmatrix}$$

Multiplying these together,

$$P_1' = PW_1 = \begin{bmatrix} 1600 & 0 & 0 & 0 \\ 0 & 1600 & 0 & 0 \\ 0 & 0 & 1 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 1600 \\ 1600 \\ 8 \end{bmatrix} \tag{14}$$

Converting back to euclidean coordinates,

$$P_1' = \begin{bmatrix} 200 \\ 200 \end{bmatrix} \tag{15}$$

Using the same process, test another point

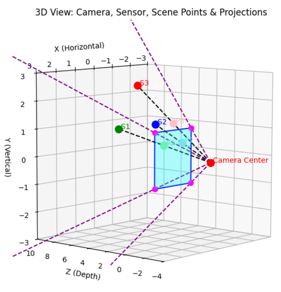
$$P_2' = P' \begin{bmatrix} -1 \\ 1 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} -200 \\ 200 \end{bmatrix} \tag{16}$$

Let's assume our camera has a sensor of 800 by 800 pixels, with the range of pixel coordinates $(-400, 400)$ in both directions. This means that P_1 and P_2 are within our camera's field of view.

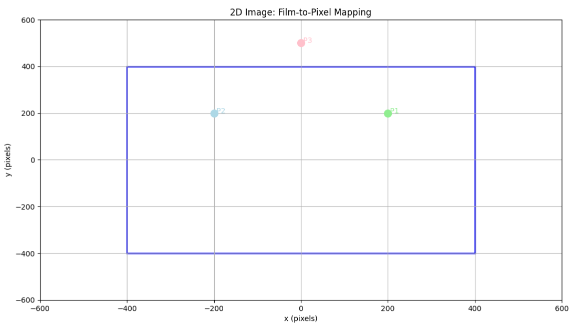
Take another point, W_3,

$$P_3' = P' \begin{bmatrix} 0 \\ 2.5 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 500 \end{bmatrix} \tag{17}$$

These pixel coordinates are outside of the camera's sensor, meaning that the camera does not capture this point!



(a) An angled view of the scene used



(b) The pixel coordinates of P_1', P_2', and P_3'

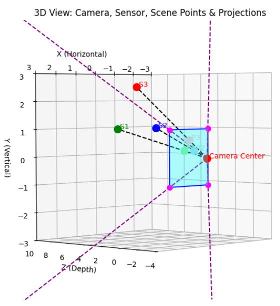
Figure 4: Pinhole camera model geometry

Let's see what happens when we decrease the focal length of the camera. Let $f = 2$. This modifies the camera matrix to

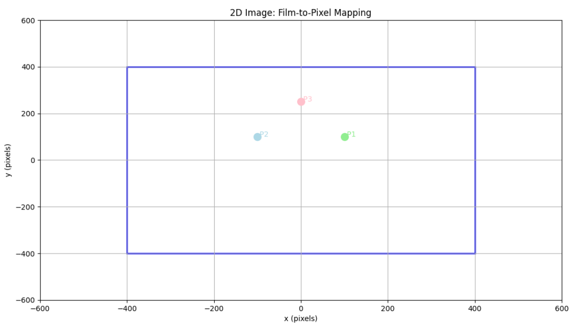
$$P = \begin{bmatrix} 800 & 0 & 0 & 0 \\ 0 & 800 & 0 & 0 \\ 0 & 0 & 1 & 3 \end{bmatrix} \tag{18}$$

Recalculating the pixel coordinates of all three points,

$$P_1' = \begin{bmatrix} 100 \\ 100 \end{bmatrix}, \; P_2' = \begin{bmatrix} -100 \\ 100 \end{bmatrix}, \; P_3' = \begin{bmatrix} 0 \\ 250 \end{bmatrix}$$



(a) Angled view of the scene, with $f = 2$



(b) Pixel coordinates with $f = 2$

Figure 5: Pinhole camera model geometry

As shown in 5b, this means that all three points are captured in the image! Anybody slightly familiar with using a camera will know this as "zooming in" or "zooming out". Indeed, increasing the focal length is exactly what "zooming in" is, while decreasing the focal length gets the opposite effect.

General Projective Cameras

Finite projective cameras are a subset of the general projective camera model. A **general projective camera** can be represented by any 3×4 matrix with rank 3. This model does not have a non-singular restriction on left 3×3 sub matrix. The requirement on rank arises because if the rank was less than 3, the projection would map to a point or line not a plane. This general projection takes the form of

$$x = PX \tag{19}$$

where again x are homogeneous pixel coordinates and X are homogeneous world coordinates. We will now explore how the camera's geometries are encoded within this matrix. As a notation P can be decomposed into the block notation $P = [M \,|\, p_4]$, with p_4 just being the 4th column of P.

Camera Center

The matrix P has a nullity of 1 because its rank, 3, is one less than the number of columns, four. Suppose the basis of the $\ker(P)$ is the 4-vector C where $PC = 0$. We will now show that C is the 3D point for the camera center represented as a homogeneous 4-vector.

First, let A be a point in 3-dimensions. Then the line between C and A can be written as the join

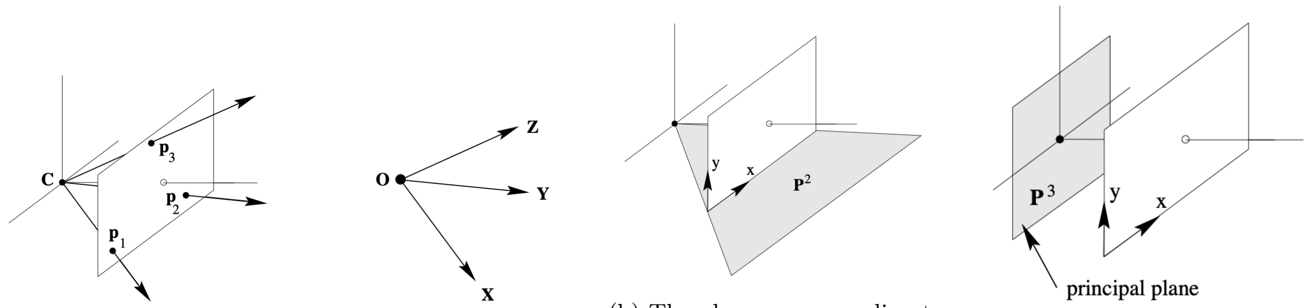
$$X(\lambda) = \lambda A + (1 - \lambda)C \quad \text{where } \lambda \in \mathbb{R}. \tag{20}$$

The join is just another way to express a line and can be rewritten in vector notation as $X(t) = C + (A - C)t$. Mapping this line under x=PX we get

$$x = PX(\lambda) = \lambda PA + (1 - \lambda)PC = \lambda PA \tag{21}$$

since $PC = 0$. So all points on the line are mapped to the same pixel point PA. This means the line must be a ray through the camera center. Since this is true for every A, it follows that C must be the homogeneous representation of the camera center. This is unsurprising since $(0, 0, 0)^\top = PC$ is undefined and the camera center is the unique point in space where the image is not undefined. The point $(0, 0, 0)^\top$ is undefined because in 2-dimensional Euclidean coordinates it is $(\frac{0}{0}, \frac{0}{0})^\top$. For a finite camera it is apparent that the camera center, $(C^\top, 1)^\top$, is the null space of P, $P = KR[I \,|\, -C]$. The result still holds when the left hand 3×3 sub matrix is singular. In this case M has a nullity of 1 and therefore there exist a 3-vector, d, such that $Md = 0$. Therefore, the camera center is located at the homogeneous coordinates $(d^\top, 0)^\top$ and its Euclidean points are at infinity. This is called an **affine projective camera**.

Column Vectors of P



(a) The 3 image points defined by p_i $i = 1, .., 3$ of the projection matrix are the vanishing points of the world coordinates axes (see appendix B).

(b) The plane corresponding to $\mathbf{P}^{2\top}$.Note the shift corresponding to the shift from camera to pixel coordinates.

(c) The principal plane is the the plane at infinity for pixel coordinates

Figure 6: Geometric interpretations for subspaces of P

The column vectors are 3-vectors with geometric meaning as particular image points. We will use the notation that the columns of P are p_i $i = 1, .., 4$. Take the x-axis of the world coordinates $D = (1, 0, 0, 0)^\top$. Then $p_1 = PD$. It follows that p_1, p_2, p_3 are the vanishing points of the x, y, and z axes of the world coordinates (see appendix B for vanishing points). The column p_4 is where the origin of the world coordinate system is mapped onto the virtual image. The origin of the world coordinate system in homogeneous coordinates is $(0,0,0,1)^\top$. A visual of this is give in Figure 6a. This should be a familiar idea since the column space is the mapping from the basis vectors $e_1, ..e_n$ to $v_1, .., v_n$.

Row Vectors of P

The rows of the projective matrix are 4-vectors that can be interpreted as planes in the world coordinate system. We will introduce the notation that the rows of P are $\mathbf{P}^{i\top}$ so that

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} = \begin{bmatrix} \mathbf{P}^{1\top} \\ \mathbf{P}^{2\top} \\ \mathbf{P}^{3\top} \end{bmatrix}. \tag{22}$$

Principal Plane. First, examine $\mathbf{P}^{3\top}$ which can be viewed as either a vector in 3-Euclidean space or a plane in the form $\mathbf{P}^{3T}X = 0$. The plane formed by the row $P^{3\top}$ consists of all points $PX = (x, y, 0)^\top$ which are imaged on the plane at infinity of the virtual image (see figure 6c and figure 10c in appendix A). If a 4-vector defining a plane is confusing, remember that we can define a plane as $Ax + By + Cz + D = 0$. Our 4-dimension in 3-Euclidean space encodes a shift which is what homogeneous vectors allow us to do (see appendix A). If C is the camera center then $PC = 0$ and $P^{3\top}C = 0$. Therefore, the camera center is part of the principal plane.

Axes Plane. Next, consider the points on the plane $P^{1\top}$. This set satisfies $P^{1\top}X = 0$ and is mapped to $PX = (0, y, W)$. Similarly $P^{2\top}$ consists of the set of points mapped to $PX = (x, 0, W)$. Both of these planes contain the center point C which is always mapped to $PC = 0$. The plane $P^{1\top}$ can be visualized as being defined by the center point, C, and the y-axis in pixel coordinates. Likewise, the plane $P^{2\top}$ is defined by C and the x-axis. The plane $P^{2\top}$ is shown in figure 6b. The axes planes' triangular geometry comes from prospective geometry and how we defined homogeneous coordinates (see see appendix A). In short, we are visualizing the plane $P^{2\top}$ in the homogeneous pixel coordinate system. In this choice of coordinates system the plane consisting of the set of points $(x, W) \in \mathbb{R}^2$. W is not a normal Euclidean coordinate it is in one sense a scalar of (x, y). As W grows, the x-axis gets longer and as it shrinks, the x-axis approaches the point at infinity corresponding to its slope of 0. This point is C, the center of the camera.

Unlike the principal plane, the axis planes are dependent on the x and y axis. This causes them to be less coupled with camera's geometry and more dependent on the orientation of the camera. The intersection of the axes plane form a line that is the back-projection from the virtual image to the origin of the world coordinate system. The camera center lies on all three of the planes and since these planes are unique (P was defined with rank 3), it must lie on the intersection. The Algebraic condition for the camera center to lie on the three planes is $PC = 0$ which is the original equation given above.

Principal Point

Recall that the principals axis (or optical axis) is the line perpendicular to the principal plane $P^{3\top}$ passing through the camera center. The principal axis intersect the virtual image at the principal point. We may determine this point as follows. The normal to a plane $\mathbf{N} = (n_1, n_2, n_3, n_4)^\top$ is the vector $(n_1, n_2, n_3)^\top$. This vector can be expressed in homogeneous coordinates as a point on the plane at infinity corresponding to its direction $(n_1, n_2, n_3, 0)^\top$. In the case of the principal plane $P^{3\top}$ this point is $\hat{P} = (p_{31}, p_{32}, p_{33}, 0)^\top$. Projecting this point onto the virtual image $P\hat{P}$ produces the principal point. Note that only the left hand 3x3 matrix $P = [M \,|\, p_4]$ is involved in this formula. This leads us to the principal point

$$p_o = Mm^{3\top} \tag{23}$$

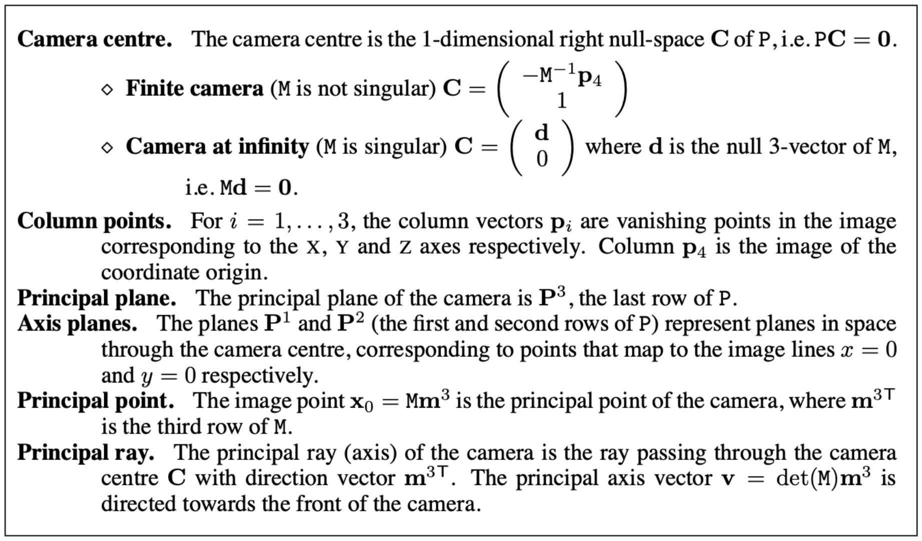
where $m^{3\top}$ is the third row of M.

Principal Axis Vector

Currently, any point not on the principal plane can be mapped to the virtual image using $x = PX$. However, in reality we know that points behind the camera cannot be seen in the image. Above, the principal point was derived using the vector for the principal plane. However, the direction of this vector is ambiguous and it is not clear whether $m^{3\top}$ or $-m^{3\top}$ points in the +x direction.

The first step to resolving this ambiguity is to go back to the camera coordinate system where the front of the camera is defined in the $+z_{cam}$ direction. The transformation from a point X_{cam} to the point x on the virtual image is given by (6), $x = P_{cam}Xcam = K[I \,|\, 0]X_{cam}$. Then $v = \det(K)k^{3\top} = f^2 m_x m_y (0,0,1)^\top$ is in the $+z_{cam}$ direction (there cannot be negative pixels per distance). Next, consider the reasonable operations that are performed on K to get to our final matrix M. K can be scaled. If $P_{cam} \rightarrow kP_{cam}$, then $v \rightarrow k^4 v$. K can also be rotated when we go from camera world to camera coordinates. Since the $\det(R) > 0$ v's sign is once again unaffected by the transformation. Revisiting our transformation in the perspective of world coordinates $P = kKR[I \mid -CR] = [M \mid p_4]$ where $M = kKR$. Therefore $v = \det(M)m^{3\top}$ points in $+z_{cam}$ direction.

Summary



Camera centre. The camera centre is the 1-dimensional right null-space $\mathbf{C}$ of $\mathtt{P}$, i.e. $\mathtt{P}\mathbf{C} = \mathbf{0}$.

◇ **Finite camera** ($\mathtt{M}$ is not singular) $\mathbf{C} = \begin{pmatrix} -\mathtt{M}^{-1}\mathbf{p}_4 \\ 1 \end{pmatrix}$

◇ **Camera at infinity** ($\mathtt{M}$ is singular) $\mathbf{C} = \begin{pmatrix} \mathbf{d} \\ 0 \end{pmatrix}$ where $\mathbf{d}$ is the null 3-vector of $\mathtt{M}$, i.e. $\mathtt{M}\mathbf{d} = \mathbf{0}$.

Column points. For $i = 1, \ldots, 3$, the column vectors $\mathbf{p}_i$ are vanishing points in the image corresponding to the X, Y and Z axes respectively. Column $\mathbf{p}_4$ is the image of the coordinate origin.

Principal plane. The principal plane of the camera is $\mathbf{P}^3$, the last row of $\mathtt{P}$.

Axis planes. The planes $\mathbf{P}^1$ and $\mathbf{P}^2$ (the first and second rows of $\mathtt{P}$) represent planes in space through the camera centre, corresponding to points that map to the image lines $x = 0$ and $y = 0$ respectively.

Principal point. The image point $\mathbf{x}_0 = \mathtt{M}\mathbf{m}^3$ is the principal point of the camera, where $\mathbf{m}^{3\mathsf{T}}$ is the third row of $\mathtt{M}$.

Principal ray. The principal ray (axis) of the camera is the ray passing through the camera centre $\mathbf{C}$ with direction vector $\mathbf{m}^{3\mathsf{T}}$. The principal axis vector $\mathbf{v} = \det(\mathtt{M})\mathbf{m}^3$ is directed towards the front of the camera.

Figure 7: Summary of properties of the projective camera. The matrix is represented in the block form $P = [M \mid p_4]$

Solving for Camera Orientation

Wilibus goes on adventure to find the Origin of the World. When he finds it he takes a picture with his nifty camera. Everyone is amazed by the beautiful picture Wilibus takes. They want to know the position and orientation and internal characteristics of his camera so they can recreate his master piece on their own adventure. Figure 8 shows Wilibus's amazing picture.

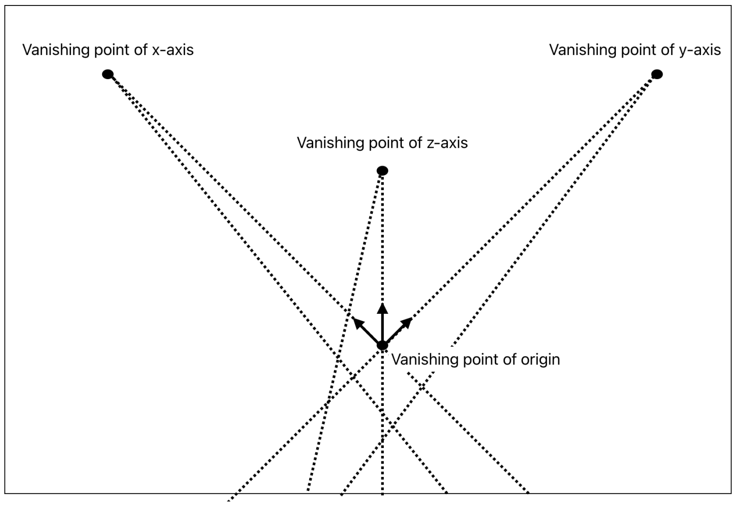

Vanishing point of x-axis
Vanishing point of y-axis
Vanishing point of z-axis
Vanishing point of origin

Figure 8: Wilibus's picture of the origin of the world. Because of perspective geometry parallel lines intersect at vanishing points (see appendix B)

Unfortunately the journey to the Origin of the World is not an easy one. In a fight with Bigfoot, Wilibus's camera was destroyed (he uploaded the photo to the cloud before hand) and it is needless to say that he forgot where he was standing when he took the picture. However, As Wilibus takes a closer look at his picture he notices that lines parallel to the x-axis, y-axis, and z-axis appear to converge at singular points on his photo. Thinking back to the linear algebra project he did in Dr. Bekyel's class, he remembers that the vanishing points for the world coordinate axes in addition to the mapping of the Origin of the World are the columns of the camera matrix. He quickly writes a Python program to find these points.

The vanishing point of the x-axis, y-axis and z-axis are located at the points $(-9,-9)$, $(3,6)$, and $(6,-\frac{2}{3})$ respectively. The Origin of the World is at the point $(-2,3)$. Rewriting these points in homogenous coordinates as column vecors he finds his camera matrix,

$$P = \begin{bmatrix} 3 & 2 & 4 & -2 \\ 3 & 4 & -1 & 3 \\ -\frac{1}{3} & \frac{2}{3} & \frac{2}{3} & 1 \end{bmatrix}. \tag{24}$$

P can be decomposed into the blockform $P = [M|t]$ where

$$M = \begin{bmatrix} 3 & 2 & 4 \\ 3 & 4 & -1 \\ -\frac{1}{3} & \frac{2}{3} & \frac{2}{3} \end{bmatrix}. \tag{25}$$

Note that M has a non-zero determinant. Using RQ decomposition (see Appendix C for the exact math), Wilibus can decompose M into the intrinsic camera matrix K and an orthogonal rotation matrix R, with

$$K = \begin{bmatrix} 4 & 2 & 3 \\ 0 & 5 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \quad R = \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{2}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{2}{3} & \frac{2}{3} \end{bmatrix} \tag{26}$$

This gives Wilibus the intrinsic camera matrix, K, which contains information about the focal length, pixel size, principle point, and skew of the camera. It also gives Wilibus the rotation part of the transformation between world and camera coordinates.

Note that R is made up of three different rotation matrices multiplied together, decomposing to $R_x(\frac{\pi}{4})$, $R_y(\frac{\pi}{9})$, and $R_z(\frac{\pi}{4})$.

To extract the translation, recall that the the camera matrix can be constructed with $P = K[R|-RC]$, implying that

$$t = -RC \tag{27}$$

As R is orthogonal (meaning $R^T = R^{-1}$), Wilibus can compute the world coordinates for the camera center, C, using

$$C = -R^T t = -\begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{2}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{2}{3} & \frac{2}{3} \end{bmatrix} \begin{bmatrix} -3 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} -\frac{1}{3} \\ -\frac{10}{3} \\ \frac{5}{3} \end{bmatrix} \tag{28}$$

Now, Wilibus has the full transformation between world and camera coordinates, so others can replicate his picture!

Conclusion: Husky Satellite Lab

Wilibus' adventure brings us to the work being done at Husky Satellite Lab (HSL).

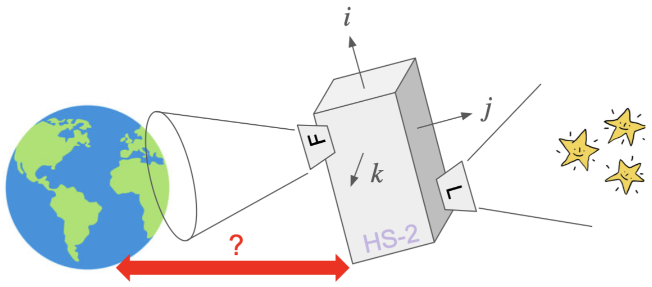


Figure 9: LOST and FOUND on HuskySat-2

HSL has received a two hundred and fifty thousand dollar (!) grant from the Department of Defense to develop the HuskySat-2 mission. HuskySat-2 will demonstrate an optical navigation package that uses star-tracking and horizon-tracking to estimate the position of the satellite. To do this estimation, HSL is developing two tools, LOST (Lost Open-Source Start Tracking) and FOUND (Found Open-Source Universal Navigation Determiner). LOST uses star-tracking to determine the attitude of the satellite, while FOUND uses images of the earth to determine the altitude of the satellite.

Fundamentally, FOUND is attempting to reverse the camera projection, going from points on a 2D image to the cameras orientation in 3D space. This is impossible unless outside information can be used. FOUND has previously used an edge detection based algorithm to find its altitude and is now exploring the viability of feature mapping. In feature mapping machine learning is used to locate and classify known points on the earth's surface, such as the edge of the planet, certain continents, or other distinctive features visible from space. Next a database is used to identify the world coordinates of these features. FOUND can then use this one coordinate to reverse-engineer the position of the satellite. Multiple coordinates and a least-squares regression method can be used to decrease error.

Let's quickly explore how combining knowledge of the camera parameters and output of LOST, attitude, with the known location of a single point can give us the position of the satellite. Assume K and R are known and that Seattle was located on the image at point (x_S, y_S). In the Earth-Centered Earth-Fixed coordinate system (ECEF), Seattle is located at approximately $X_S = -2300489$m, $Y_S = -3460794$m, and $Z_S = 4881760$m. Using (12) we get

$$\begin{bmatrix} x_S \\ y_S \\ 1 \end{bmatrix} = KR[I \mid -C_?] \begin{bmatrix} X_S \\ Y_S \\ Z_S \\ 1 \end{bmatrix} = KR \begin{bmatrix} X_S \\ Y_S \\ Z_S \end{bmatrix} - \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix} \tag{29}$$

which gives us a final position of

$$\begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix} = KR \begin{bmatrix} X_S \\ Y_S \\ Z_S \end{bmatrix} - \begin{bmatrix} x_S \\ y_S \\ 1 \end{bmatrix} \tag{30}$$

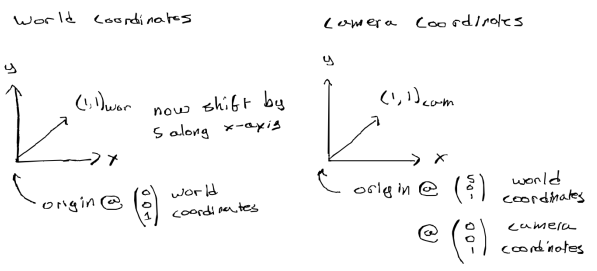
in ECEF coordinates.

Appendix

A Homogeneous Coordinates

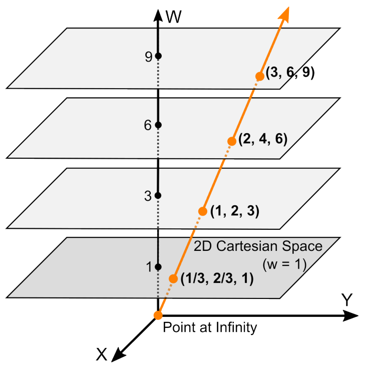
Why do we have homogeneous coordinates? Common sense tells us that two parallel lines cannot intersect, which is completely true in Euclidean geometry. However, in projective geometry two parallel lines can intersect and they do so at the vanishing point at ∞ (see appendix B). This is shown by the railroad lines in figure 10b intersecting at the horizon.

In 2-Euclidean coordinates the point at (∞, ∞) is meaningless to fix this problem we introduce homogeneous coordinates. Homogeneous coordinates are the same as Euclidean coordinates except with additional points added. These points are called points at infinity and there is a unique one for every slope. To construct homogeneous coordinates given a 2-Euclidean coordinates $(x, y)^\top$ add a non-zero number W to form the triple $(xW, yW, W)^\top$. This is the homogeneous representation of that point. For example, $(1, 2)^\top$ becomes $(1, 2, 1)^\top$ or $(2, 4, 2)$ in homogeneous coordinates. Thus a single point in Euclidean coordinates can be represented by infinite homogeneous coordinates (see figure 10c).


World Coordinates
y
x
$(1,1)_{wor}$
now shift by 5 along x-axis
origin @ $\begin{pmatrix}0\\0\\1\end{pmatrix}$ world coordinates
Camera Coordinates
y
x
$(1,1)_{cam}$
origin @ $\begin{pmatrix}5\\0\\1\end{pmatrix}$ world coordinates
@ $\begin{pmatrix}0\\0\\1\end{pmatrix}$ camera coordinates

(a) A translation of the origin using homogeneous coordinates

(b) In perspective geometry parallel lines intersect at ∞, the vanishing point



(c) Euclidean coordinates as a layer in homogeneous coordinates

Figure 10: Visualization of homogeneous coordinates

In 2-Euclidean coordinates the point $(0, 0)^\top$ corresponds to the origin which in homogeneous would be represent as $(0, 0, 1)^\top$. Note that $(0, 0, W)^\top$ where W is non-zero also corresponds to the origin in 2-Euclidean coordinates. When Z is not 1 then the coordinate is scaled by a factor of W and thus we use $(0, 0, 1)^\top$ as the homogeneous coordinate for the origin in 2-Euclidean. The The equation of a line through the origin can be written as $nx + my = 0$ where n and m are not zero and $\frac{-n}{m}$ is the slope. Then parameterizing the equation we get $x = mt$, $y = -nt$, and $W = \frac{1}{t}$. The the coordinates of a point on the line can be written as $(\frac{m}{W}, \frac{-n}{W})^\top$. In homogeneous coordinates this becomes $(m, -n, W)^\top$. As we travel down the line $nx + my = 0$ and t goes to infinity W approaches zero. Thus we define $(m, -n, 0)^\top$ as the point at infinity of the line $nx + my = 0$ with slope $\frac{-n}{m}$.

As a quick exercise we will prove that parallel lines in 2-Euclidean space can converge in homogeneous coordinates. Consider the following linear system in 2-Euclidean

$$\begin{cases} Ax + By + C = 0 \\ Ax + By + D = 0. \end{cases} \tag{31}$$

The only solution to the equation is when the lines are parallel, $C = D$. Therefore the lines never intersect. We can introduce homogeneous coordinates by switching (x, y) to $(\frac{x}{W}, \frac{y}{W})$

$$\begin{cases} A\frac{x}{w} + B\frac{y}{w} + C = 0 \\ A\frac{x}{w} + B\frac{y}{w} + D = 0 \end{cases} \Rightarrow \begin{cases} Ax + By + Cw = 0 \\ Ax + By + Dw = 0. \end{cases} \tag{32}$$

Now there is a solution to the equation, $w = 0$. The result agrees with our setup. Parallel lines intersect at infinity. This point is called the vanishing point (see appendix 11).

Suppose we want to perform a translation. Linear systems require that the origin is always mapped to the origin. With this requirement it would seem that it is impossible to describe a translation using linear algebra. However homogeneous coordinates once again save the day. Consider the 2-Euclidean example in figure 10a. The origin in our original coordinates is the basis vector $(0, 0, 1)^\top$. Say we want to shift our go from our original coordinate system to one with origin 5 units in the +x-axis direction. Then our new coordinate system x would have coordinates $x_{old} = (5, 0)^\top$ with respect to the old coordinate system. We could relate the original coordinate system X to the new coordinates by

$$x = X - x_{old} \tag{33}$$

We can view this as the Euclidean basis vectors of X being unchanged and the homogeneous "origin" vector being shifted. With homogeneous coordinates this linear transformation can be represented as

$$x_h = \begin{bmatrix} I_{2x2} & -(5,0)^\top \\ 0 & 1 \end{bmatrix} X_h \tag{34}$$

where h denotes homogeneous coordinates. Let's check if this transformation does what we want it to do. Let's choose the vector $(1, 1)^\top$ in our old coordinate system. In our new coordinate system shifted 5 units to the right this vector should be $(-4, 1)^\top$.

$$x_h = \begin{bmatrix} I_{2x2} & -(5,0)^\top \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -4 \\ 1 \\ 1 \end{bmatrix} \tag{35}$$

Note that our transformation is 3x3 matrix with full rank. Therefore we can find the inverse which should send points in our new coordinate system back to our old coordinate system. The inverse of our matrix turns out to be quite simple

$$\begin{bmatrix} I_{2\times 2} & (5,0)^\top \\ 0 & 1 \end{bmatrix}. \tag{36}$$

Looking at the 3rd column of the inverse we can see that it shifts the origin 5 units in the other direction. Mapping the coordinates $(-4, 1)^\top$ back to the old coordinate system we get

$$x_h = \begin{bmatrix} I_{2x2} & (5,0)^\top \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -4 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \tag{37}$$

our original vector.

B Vanishing points

Vanishing points are the points where parallel lines intersect on the image pane. This is caused by prospective geometry (see appendix A). An example of a vanishing point can be seen in figure 10b and figure 11. where as two parallel lines stretch to infinity they are imaged as an intersection on the image plane.

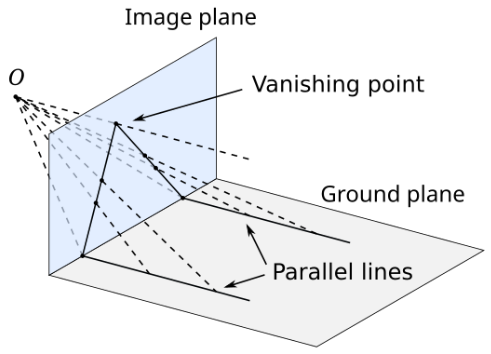


Figure 11: Visualization of how two parallel lines intersect on the image plane at a vanishing point as they go to infinity

C RQ Decomposition

RQ decomposition is an algorithm used to decompose a matrix into a matrix R and a matrix Q, where R is upper-triangular and Q is a rotational matrix.

Here, we take a 3×3 matrix M, given by

$$M = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

which we will use in the context of RQ decomposition and its related transformations. Specifically, to do RQ decomposition, we will take A and turn it into an upper triangular matrix along with some rotational matrix R. For a reason unbeknownst to the author of this paper, some people find it confusing that R is an upper-triangular matrix and Q is a rotational matrix. So, we will rename our upper triangular matrix as K (to match with standard conventions of camera matrices) and our rotational matrix as R (so that the matrix's letter matches with the first letter of the word *rotation* — you're welcome).

To transform matrix M into K and R, we multiply by three different individual matrices, which, by convention, we will denote as Q_x, Q_y, and Q_z. Each of these matrices are given by the formulas shown below. Before we jump into these, however, we must define the key parts of them. As such, we let

$$c = \cos(\theta) = \frac{-a_{i,j+1}}{\sqrt{a_{i,j+1}^2 + a_{ij}^2}}, \quad s = \sin(\theta) = \frac{a_{ij}}{\sqrt{a_{i,j+1}^2 + a_{ij}^2}}$$

where a_{ij} represents the value that you are trying to make equal to zero following the given rotational multiplication. As an example, if we try to set the entry a_{32} to 0, we need to solve the equation

$$ca_{32} + sa_{33} = 0,$$

with

$$c = \frac{-a_{33}}{\sqrt{a_{33}^2 + a_{32}^2}}, \quad s = \frac{a_{32}}{\sqrt{a_{33}^2 + a_{32}^2}}$$

Also of use is the trigonometric identity from the unit circle:

$$c^2 + s^2 = 1,$$

since $c = \cos(\theta)$ and $s = \sin(\theta)$.

Rotation Matrices

$$Q_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c & -s \\ 0 & s & c \end{bmatrix}, \quad Q_y = \begin{bmatrix} c & 0 & s \\ 0 & 1 & 0 \\ -s & 0 & c \end{bmatrix}, \quad Q_z = \begin{bmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

By multiplying M by all three of these matrices, and remembering to redefine your c and s values for each individual matrix (i.e., use c_x, s_x, c_y, s_y, c_z, and s_z), you decompose the function into K and R, with:

$$R = Q_x^T Q_y^T Q_z^T.$$

References

[1] A. Sriram, ”Camera Projections,” Medium, Available: https://medium.com/@abhisheksriram845/camera-projections-43227389e55d.

[2] J. A. Christian, “A Tutorial on Horizon-Based Optical Navigation and Attitude Determination With Space Imaging Systems,” IEEE Access, vol. 9, pp. 19819–19853, Jan. 2021, doi: 10.1109/ACCESS.2021.3051914.

[3] K. Hata and S. Savarese, ”CS231A Course Notes 1: Camera Models,” Available: https://web.stanford.edu/class/cs231a/course_notes/01-camera-models.pdf.

[4] K. Kitani, ”Lecture 11.1: Camera Matrix,” Carnegie Mellon University, Available: https://www.cs.cmu.edu/~16385/s17/Slides/11.1_Camera_matrix.pdf.

[5] R. Hartley and A. Zisserman, ”Multiple View Geometry in Computer Vision”, 2nd edition. Cambridge, United Kingdom, Cambridge University Press, 2004, pp 154-162.