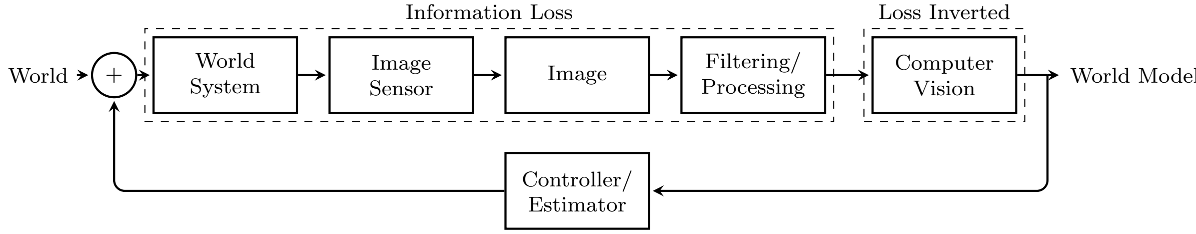
Computer Vision Notes [1]

Eric Squires February 16, 2013

Contents

1 Introduction

- Computer Vision - detection of 3d properties (geometric, material) from 2d images (inverse problem)
  - Geometric Properties - size, shape, location
  - Material Properties - radiance, color, texture, material composition
- Computer Vision is not image processing or pattern recognition
  - Pattern Recognition - classifies patterns into a finite number of categories (e.g., is there a person in the picture?)
  - Image processing - producing new image from an old one (often a precursor to computer vision)
- Ways of Approaching Computer Vision: high vs low level, biological vs synthetic
  - Biological - tends to be more complicated/low level
- Block Diagram view
  - Controller - gets something to work (thing in actual world) how we want it to work (model)
  - Estimator - figures out how something (model) works by observing its behavior (actual world)

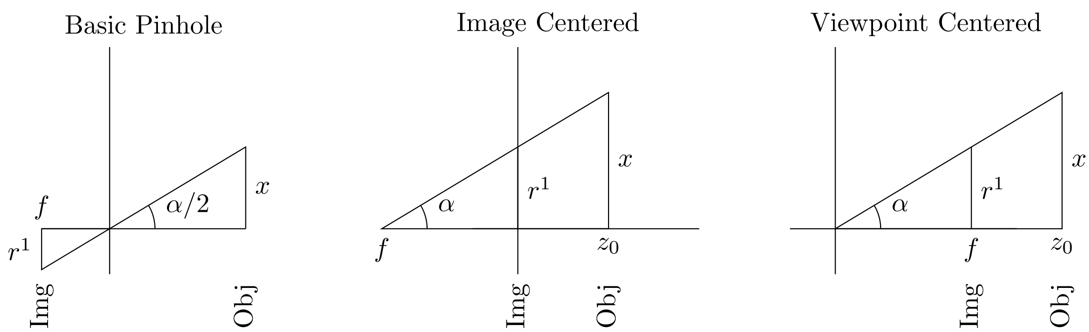


2 Image Formation and Sensing

Steps for image formation and sensing

2.1 Image Formation

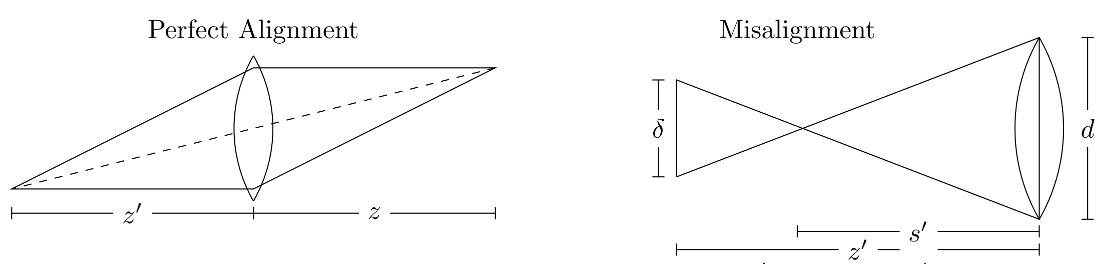
2.1.1 Models of Projection



$$r^1 = -\frac{fx}{z_0} \qquad r^1 = \frac{fx}{z_0} \qquad r^1 = \frac{fx}{f+z_0}$$

- Center of Projection - location of pinhole
- Pinhole problems - does not let in enough light, diffraction (reason for using lenses)
- If $f \gg z$ (e.g., microscope), then $r^1 = \frac{fx}{f+z} \approx x$ (use viewpoint-centered)
- If $f \gg \Delta z$ (e.g., a wall), then $r^1 = \frac{fx}{z+\Delta z} \approx \frac{fx}{z} = mx$ where $\boxed{m = \frac{f}{z}}$

2.1.2 Lenses



$$\frac{1}{z'} + \frac{1}{z} = \frac{1}{f} \qquad \delta = d\left|\frac{1-f/s}{1-f/z} - 1\right|$$

To derive the latter equation, define the following:

- z' - actual distance of lens to image plane
- s' - ideal distance of lens to image plane
- z - actual distance to the object
- s - ideal distance to the object

Then by similar triangles, $s'\delta = d(z' - s')$. Then

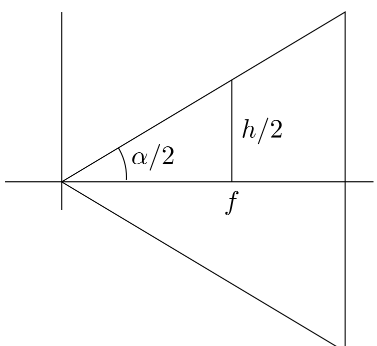
$$
\begin{aligned}
\delta &= \frac{d(z' - s')}{s'} \\
&= \frac{d|z' - s'|}{s'} && \text{(whether } s' \text{ is too far or too close does not not matter)} \\
&= d\left|\frac{z'}{s'} - 1\right| \\
&= d\left|\frac{fz/(z-f)}{fs'/(s-f)} - 1\right| && \text{(From lens equation: } \frac{1}{z'} + \frac{1}{z} = \frac{1}{f} \Rightarrow z' = \frac{fz}{z-f} \text{ and } s' = \frac{sz}{s-f}\text{)} \\
&= d\left|\frac{z(s-f)}{s'(z-f)} - 1\right| \\
&= d\left|\frac{zs(s-f/s)}{zs(1-f/z)} - 1\right| \\
&= d\left|\frac{1-f/s}{1-f/z} - 1\right|
\end{aligned}
$$

Other interesting (related) points:

- Aperture - a smaller d means less blurring but also less light coming in.
- Depth of Field - range of distances over which objects are focused sufficiently well (e.g., $s \in [z_{min}, z_{max}]$)
- Resolution - higher resolution means lower depth of field (less tolerance for δ)

2.2 Image Sensing

2.2.1 Field of View



Viewpoint Centered View

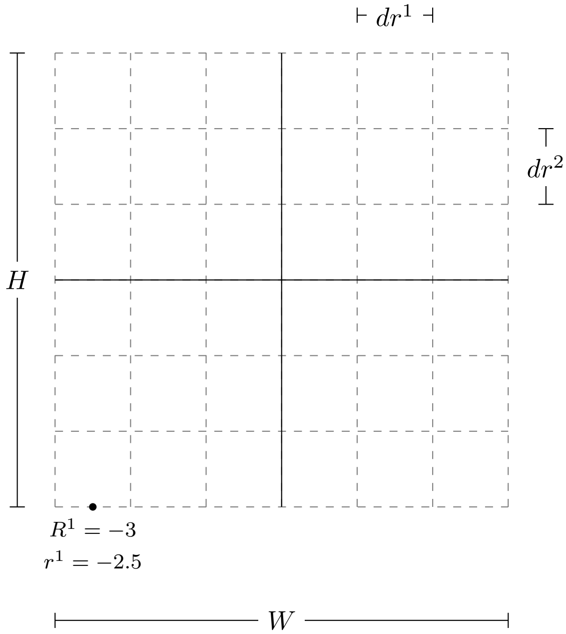
$$\alpha = 2\arctan\left(\frac{h}{2f}\right) = \text{Field of View}$$

2.2.2 Quantization

Relevant terms[1]:

- R^1 - discrete horizontal position (i.e., the pixel #)
- r^1 - continuous horizontal position (i.e., a measurement with a ruler)
- dr^1 - width of a pixel
- W - total image plane width (in pixels)

[1] substitute superscript "2" for vertical coordinates



$$R^1 = \lfloor r^1/dr^1 \rfloor + \frac{W}{2} \ (W \text{ even})$$

$$R^1 = \lfloor r^1/dr^1 \rfloor + \frac{W-1}{2} \ (W \text{ odd})$$

2.2.3 Transformations

Terminology

1. $T_{BC}^W = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$ - translation from points B to C in the W (world) reference frame[2]
2. $R_C^W = \begin{pmatrix} | & | & | \\ \hat{x}_W & \hat{y}_W & \hat{z}_W \\ | & | & | \end{pmatrix}$ - rotation from reference frame W to C (note: $R^T R = I$, $|R| = 1$). To create R_W^C - plot W and C coordinate frames on top of each other. $\hat{x}_c$ is the location of the world's x-axis[3] from the perspective of the C frame (see applications below).
3. $g_C^W = \left(\begin{array}{c|c} R_C^W & T_{WC}^W \\ \hline 0 & 1 \end{array}\right)$ - transformation matrix from homogeneous coordinate in reference frame C to same point in W [4]
4. q_A^W - point A in reference frame W

[2] i.e., $q_B^W = T_{AB}^W + q_A$

[3] (x,y,z)

[4] i.e., $q_A^W = \left(\frac{p_A^W}{1}\right) = g_C^A q_A^C$

Transformation Summary[2]

Name	Equation	Preserves...	Notes
Translation	$\begin{pmatrix} I & t \\ \mathbf{0}^T & 1 \end{pmatrix}$	angles, lengths, parallel	
Rotation	$\begin{pmatrix} R & \mathbf{0}^T \\ \mathbf{0} & 1 \end{pmatrix}$	angles, lengths, parallel	x, y, z orthonormal
Scaling	$\begin{pmatrix} \alpha I & \mathbf{0}^T \\ \mathbf{0} & 1 \end{pmatrix}$	angles, parallel	
Shear	$\begin{pmatrix} 1 & a & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	parallel	
Affine	$\begin{pmatrix} A & \mathbf{0} \\ \mathbf{0}^T & 1 \end{pmatrix}$	parallel	A combines rotation, shear, scale
Projective	$\begin{pmatrix} A & \boldsymbol{t} \\ \boldsymbol{v} & 1 \end{pmatrix}$	straight lines	combines all of the above

3D to 2D Projection [2]: Comparing Orthographic, Para-Perspective, and Perspective Projection

- Orthographic[5] - removes the z component. Good approximation when $f \gg z$ or $f \gg \Delta z$.

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

- Para-Perspective - projects on line along line of sight to object center then scales

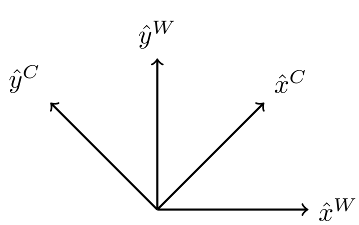
$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{13} \\ a_{13} & a_{13} & a_{13} & a_{13} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Applications

- Finding R_W^C: Write the coordinates of the world unit x-axis in terms of camera coordinates (same for y, and z). Then

$$R_W^C = \begin{pmatrix} | & | & | \\ \hat{x}_W & \hat{y}_W & \hat{z}_W \\ | & | & | \end{pmatrix}$$

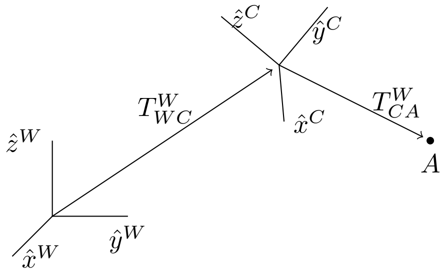
Here is a simple example for finding R_W^C. Given the below picture, we want $\begin{pmatrix} 1 \\ 0 \end{pmatrix}_W \Rightarrow \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}_C$ and $\begin{pmatrix} 0 \\ 1 \end{pmatrix}_W \Rightarrow \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}_C$



$$R_W^C = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ \text{‘} - 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ \text{‘}0 & 0 & 1\text{‘} \end{pmatrix}$$

[5] x and y may also be scaled

- Transform point from perspective of camera to perspective of world



$$q_A^W = R_C^W q_A^C + T_{WC}^W = T_{CA}^W + T_{WC}^W$$

- Turning affine operations into linear ones with homogeneous coordinates

$$q_A^W = \left(\begin{array}{c} p^W \\ \hline 1 \end{array}\right) = \left(\begin{array}{ccc|c} & & & 0 \\ & R_C^W & & 0 \\ & & & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right)\left(\begin{array}{c} p^C \\ \hline 1 \end{array}\right) + \left(\begin{array}{c} T_{WC}^W \\ \hline 0 \end{array}\right)$$

$$= \left(\begin{array}{ccc|c} & & & 0 \\ & R_C^W & & 0 \\ & & & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right)\left(\begin{array}{c} p^C \\ \hline 1 \end{array}\right) + \left(\begin{array}{ccc|c} & 0 & & T_{WC}^W \\ \hline 0 & 0 & 0 & 1 \end{array}\right)\left(\begin{array}{c} p^C \\ \hline 1 \end{array}\right)$$

$$= \left(\begin{array}{ccc|c} & R_C^W & & T_{WC}^W \\ \hline 0 & 0 & 0 & 1 \end{array}\right)\left(\begin{array}{c} p^C \\ \hline 1 \end{array}\right) = g_C^W q_A^C$$

- Finding $\left(g_C^W\right)^{-1}$:[6]

Inverting the formula from Bullet #1:	$q_A^C = (R_C^W)^{-1} q_A^W - (R_C^W)^{-1} T_{WC}^W$
Direct use of formula from Bullet #1:	$q_A^C = R_W^C q_A^W + T_{CW}^C$
Combining these two gives:	$R_W^C = (R_C^W)^{-1}$ and $T_{CW}^C = -(R_C^W)^{-1} T_{CW}^W$

Using this result as well as the conclusion from bullet #3 gives a new form for $(g_C^W)^{-1}$:

$$q_A^C = g_W^C q_A^W = \left(\begin{array}{ccc|c} & R_C^W & & T_{WC}^W \\ \hline 0 & 0 & 0 & 1 \end{array}\right) q_A^W = \left(\begin{array}{ccc|c} & (R_W^C)^{-1} & & -(R_C^W)^{-1} T_{CW}^W \\ \hline 0 & 0 & 0 & 1 \end{array}\right) q_A^W = (g_C^W)^{-1} q_A^W$$

2.3 Going from world coordinates to camera pixels: Ψ

2.3.1 Deriving the basic form of Ψ

3 steps

1. Image Sensing ($q^w \to q^c$): Get points in camera frame (i.e., $q^c = g_w^c q^w$)

$$q^c = \left(\begin{array}{c} p^c \\ \hline 1 \end{array}\right) = g_w^c q^w = \left(\begin{array}{c|c} R & T \\ \hline 0 & 1 \end{array}\right) q^w$$

[6]remember that because R is orthonormal, $R^T = R^{-1}$

2. Projection ($q^c \to \boldsymbol{r}$ where $\boldsymbol{r}$ is continuous): Apply perspective projection equations (i.e., $r = (fx^c/z^c, fy^c/z^c)^T$)[7]

$$\boldsymbol{r} = \begin{pmatrix} r^1 \\ r^2 \\ 1 \end{pmatrix} = \begin{pmatrix} fx^c/z^c \\ fy^c/z^c \\ 1 \end{pmatrix} \sim \begin{pmatrix} fx^c \\ fy^c \\ z^c \end{pmatrix} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x^c \\ y^c \\ z^c \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} q^c$$

Now combine this information with that from step 1

$$\boldsymbol{r} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \left(\begin{array}{c|c} R & T \\ \hline 0 & 1 \end{array}\right) q^w = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \left(\begin{array}{c|c} R & T \end{array} \right) q^w = \Psi_1 \left(\begin{array}{c|c} R & T \end{array} \right) q^w$$

2 other forms: $\Psi_1 \left(R|T\right) q^w = \left(\Psi_1 R_W^C | \Psi_1 T_W^C\right) q^w = \left(\Psi (R_c^w)^T | - \Psi (R_c^w)^T T_c^w\right) q^w$

3. Quantize Signal - translate,[8] scale, skew to correct camera abnormalities (e.g., center of focus is not in center of camera, etc). Then round in order to place into buckets

Scaling: $\begin{pmatrix} \alpha & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$ Translation: $\begin{pmatrix} 1 & 0 & t_1 \\ 0 & 1 & t_2 \\ 0 & 0 & 1 \end{pmatrix}$ Skew: $\begin{pmatrix} 1 & \delta & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

When we multiply these equations together, we get a matrix that does all three operations: $\begin{pmatrix} \alpha & \delta & t_1 \\ 0 & \alpha & t_2 \\ 0 & 0 & 1 \end{pmatrix}$.

Multiplying this matrix by what we found in the previous step gives the final Ψ[9]:

$$\begin{pmatrix} \alpha & \delta & t_1 \\ 0 & \alpha & t_2 \\ 0 & 0 & 1 \end{pmatrix} \Psi_1 \left(\begin{array}{c|c} R & T \end{array} \right) q^w = \begin{pmatrix} \alpha & \delta & t_1 \\ 0 & \alpha & t_2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \left(\begin{array}{c|c} R & T \end{array} \right) q^w$$

$$= \begin{pmatrix} \alpha f_1 & \delta & t_1 \\ 0 & \alpha f_2 & t_2 \\ 0 & 0 & 1 \end{pmatrix} \left(\begin{array}{c|c} R & T \end{array} \right) q^w = \Psi \left(\begin{array}{c|c} R & T \end{array} \right) q^w$$

Note: The rounding step (e.g., floor, ceil) is a non-linear function not captured by Ψ.

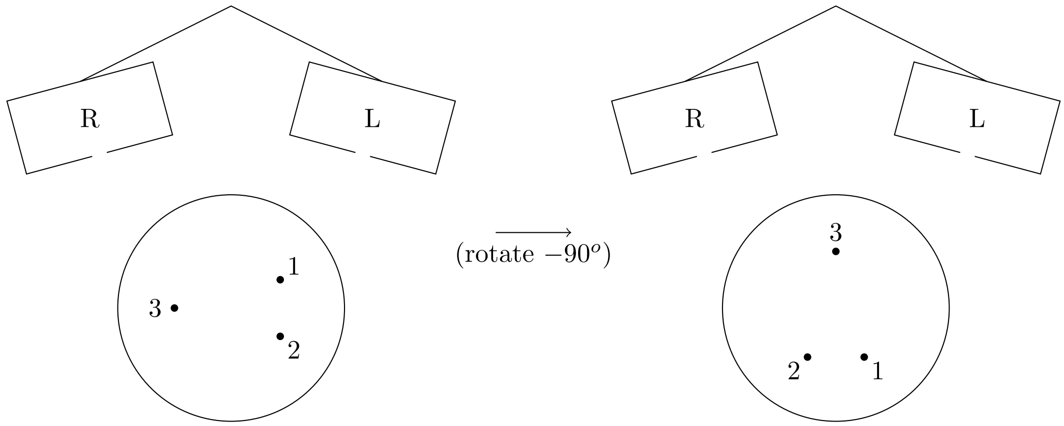
2.3.2 Special Topic: Linearizing a Matrix

Suppose you have a stereo camera at $t = t_1$ taking a picture of some object. You then rotate the object and want to know how how to transform points on the object at t_1 to the same points at t_2. Note that the object is rigid so the transformation matrix will be the same for all points. Only from observing points, what is the transformation from t_1 to t_2?

[7] note that $(x^c, y^c, z^c)^T = q^c$.

[8] to make the middle the origin of the xy axis, t_1 and t_2 are often set to $W/2$ and $H/2$ respectively. Note that because of projection, translation is just for x and y. There is no z.

[9] note that when the buckets on the camera image sensor are not equal, we get different f_1 and f_2



Consider some point q_i[10] for which we have its position at both t_1 and t_2. Then:

$$\begin{pmatrix} x_i(t_2)\\ y_i(t_2)\\ z_i(t_2)\\ 1\end{pmatrix} = q_i(t_2) = g q_i(t_1) = \left(\begin{array}{c|c} R & T \\ \hline 0 & 1\end{array}\right)\begin{pmatrix} x_i(t_1)\\ y_i(t_1)\\ z_i(t_1)\\ 1\end{pmatrix} = \begin{pmatrix} R_{11}x_i(t_1) + R_{12}y_i(t_1) + R_{13}z_i(t_1) + T_1\\ R_{21}x_i(t_1) + R_{22}y_i(t_1) + R_{23}z_i(t_1) + T_2\\ R_{31}x_i(t_1) + R_{32}y_i(t_1) + R_{33}z_i(t_1) + T_3\\ 1\end{pmatrix}$$

Note that we can write each component in $q(t_2)$ as follows (the first component is shown, but the 2^nd and 3^rd components can be made by moving the $\mathbf{0}^T$ around:

$$x_i(t_2) = \begin{pmatrix} x_i(t_1) & y_i(t_1) & z_i(t_1) & 1\end{pmatrix}\begin{pmatrix} R_{11}\\ R_{12}\\ R_{13}\\ T_1\end{pmatrix} = \begin{pmatrix} x_i(t_1) & y_i(t_1) & z_i(t_1) & 1 & \mathbf{0}^T_{1x4} & \mathbf{0}^T_{1x4}\end{pmatrix}\begin{pmatrix} R_{11}\\ R_{12}\\ R_{13}\\ T_1\\ \mathbf{0}^T_{4x1}\\ \mathbf{0}^T_{4x1}\end{pmatrix}$$

Thus, we can write the original equation as follows:

$$q_i(t_2) = \begin{pmatrix} x_i(t_2)\\ y_i(t_2)\\ z_i(t_2)\\ 1\end{pmatrix} = \begin{pmatrix} x_1(t_1) \quad y_1(t_1) \quad z_1(t_1) \quad 1 & \mathbf{0}^T_{1x4} & \mathbf{0}^T_{1x4}\\ \mathbf{0}^T_{1x4} & x_1(t_1) \quad y_1(t_1) \quad z_1(t_1) \quad 1 & \mathbf{0}^T_{1x4}\\ \mathbf{0}^T_{1x4} & \mathbf{0}^T_{1x4} & x_1(t_1) \quad y_1(t_1) \quad z_1(t_1) \quad 1\end{pmatrix}\begin{pmatrix} R_{11}\\ R_{12}\\ R_{13}\\ T_1\\ R_{21}\\ R_{22}\\ R_{23}\\ T_2\\ R_{31}\\ R_{32}\\ R_{33}\\ T_3\end{pmatrix}$$

There is only one problem—there are three equations and 12 unknowns. Fortunately, there is a simple solution. Because the object is rigid, every point in the object shares the same transformation matrix g. Thus, rather than using just one point in the above equation, we can augment it with extra points.[11]Thus, we are left with the familiar $b = Ax$ form and can solve for x, where x represents a linearized transformation matrix g. The final step resizes the linearized matrix to the original square version.

[10]There are only 3 points shown in the picture but there could obviously be many more.

[11]Note that in this case there is no projection so there is no information lost (i.e., the z-information is still intact). Thus, we only need to use 4 points. However, when projection is involved (see below), we will need 6 points because we will only have x and y information.

2.3.3 Camera Calibration: Solving for Ψ

Camera calibration involves identifying camera parameters by taking a picture of a scene where intrinsic calibration solves for Ψ and extrinsic calibration solves for $(R|T)$. In section 2.3.2, a system of the form $q^C = (R|T)q^W$ was solved for $(R|T)$ by linearizing the matrix $(R|T)$. Now multiply both sides by Ψ to get the following:

$$\boldsymbol{r} = \Psi q^C = \Psi(R|T)q^W$$

Although Ψ was derived in section 2.3.1, people often refer to it in other ways (for reasons listed below):

$$\boldsymbol{r} = \Psi \left(\begin{array}{c|c} R & T \end{array} \right) q^w = Dq^w = \left(\begin{array}{c|c} \Psi R & \Psi T \end{array} \right) q^w = \left(\begin{array}{c|c} M & \nu \end{array} \right) q^w$$

- Reason for D - In the worst case, $\Psi \left(\begin{array}{c|c} R & T \end{array} \right)$ has 17 unknowns (5 for Ψ, 9 for R, 3 for T), but using the D form, this comes down to only 12 unknowns. To solve for D (see section 2.3.2), arrange $\boldsymbol{r}_i = Dq_i^w$ into one of the following forms:

  $$0 = \boldsymbol{r}_i \times Q(q_i)\boldsymbol{d} \qquad 0 = \hat{r}_i Q(q_i)\boldsymbol{d}$$

  where

  $$\hat{a} = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix} \tag{1}$$

  and $Q(q_i)$ is a matrix after the linearizing process of section 2.3.2. From here, you can solve for D in one of two ways:

  – Singular Value Decomposition - given some $m \times n$ matrix A, the `svd` factors A so that

    $$A = U\Sigma V^T$$

    where U and V^T are orthonormal.[12] Then

    $$AV = U\Sigma$$

    or looking at just a column at a time gives:

    $$A\boldsymbol{v}_i = \sigma_i \boldsymbol{u}_i$$

    Assuming the problem has a solution, it amounts to finding some $\boldsymbol{v_i}$ in the null space of A. Fortunately, since we have provided enough points, it is solvable and there should be some $\sigma_k \approx 0$.[13] Then

    $$A\boldsymbol{v}_k = 0$$

    Remember that A stands for $\hat{r}_i Q(q_i)$ and v_k is the $\boldsymbol{d}$ we are looking for. Putting all this more colloquially, $\boldsymbol{d} = \boldsymbol{v}_k$ will always be the right-most vector of V so just run the following code: `[U S V] = svd(Q); V(:,end);`. Then reshape the vector to a $3x4$. Remember at this point though that $\Psi \left(\begin{array}{c|c} R & T \end{array} \right)$ and Ψ may or may not have a bottom right element of 1. In case it is not 1, divide D by $\sqrt[3]{\det(D)}$.

  – Pseudo-Inverse: `d = pinv(A)` or `d = R \ Q` in MATLAB[14] and do the rescaling in the above bullet.

  The downside of using the D matrix is that it needs to be recalibrated every time you move the camera.

[12] orthonormal implies $U^{-1} = U^T$

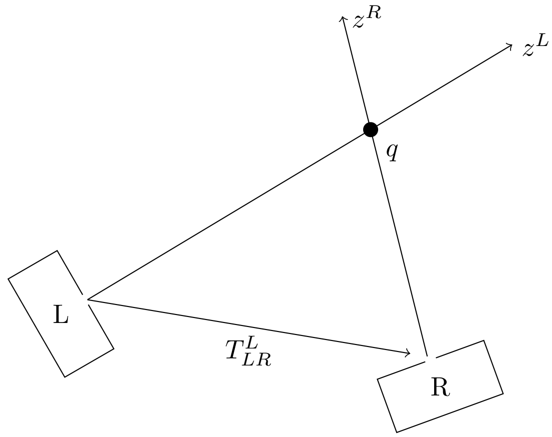
[13] The `svd` will order the singular values from highest (top left) to lowest (bottom right). Also, $k = \min(m, n)$.

[14] This method is quite sensitive to floating point arithmetic so it does not work in practice particularly well.

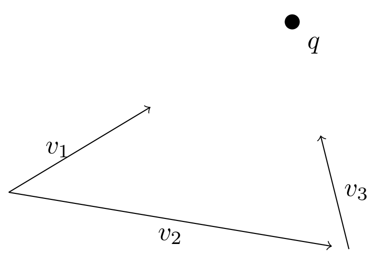
2.4 Stereo

2.4.1 Epipolar Lines

Suppose you have a stereo rig as shown below:



Consider some point projected onto camera L and note that there are an infinite number of points in the world that could have projected to this point. This is z^L in the picture. Similarly, find the same point in camera R and let z^R be the ray representing the set of points that could have projected to this point. Also, assume you know $g^L_R = \left(\begin{array}{c|c} R^L_R & T^L_R \\ \hline 0 & 1 \end{array}\right)$. Note also the following geometric relations:



Because three points determine a plane, we can write

$$0 = (v_3 \times v_2) \cdot v_1$$

and referencing equation (1), we can rewrite this as:

$$\begin{aligned} 0 &= (\hat{v}_3 v_2) \cdot v_1 \\ &= v_1 \cdot \hat{v}_3 v_2 && \text{(dot product is commutative)} \\ &= v_1^T \hat{v}_3 v_2 && \text{(definition of dot product)} \end{aligned}$$

When we compare the two figures in this section, we see an obvious correspondence between z^L, T^L_{LR},[15] z^R and v_1, v_2, v_3. There is only one slight change we need to make—we need to rotate z^R into z^L coordinate frame to create z^R. Thus, we are left with:

$$0 = (z^L)^T (\hat{T}^L_{LR} R^L_R) z^R = (z^L)^T E z^r \tag{2}$$

We can now find the set of points z^R that are in the null space of $z^L E$.

[15] $\hat{T} = \begin{pmatrix} 0 & -T^3 & T^2 \\ T^3 & 0 & -T^1 \\ -T^2 & T^1 & 0 \end{pmatrix}$ will be used below

2.4.2 Essential vs. Fundamental Matrix

The Essential matrix was derived in the previous section. To see how the Fundamental Matrix arises, note that $r^L = \Psi z^L$ so that $z^L = \Psi^{-1} r^L$. Then $(z^L)^T = (r^L)^T \Psi^{-T}$.[16] A similar process yields $z^R = \Psi^{-1} r^r$. We can plug these results into equation (2) to find F:

$$0 = (r^L)^T \Psi^{-T} (\hat{T}^L_{LR} R^L_R) \Psi^{-1} r^R = (r^L)^T F r^R$$

This will be a line in the right camera's image. Rather than searching the whole image, we can search for the corresponding point on the line.

To see why this is the case, let $w^T = (r^L)^T F$. Then

$$0 = w^T r^R = w^1 r^{R_1} + w^2 r^{R_2} + w^3$$

and note that it has the familiar form of a line: $ax + by + c = 0$.

Before contrasting the two matrices, it is helpful to summarize their forms. [17]

Matrix	Formula	Image Plane	Ray Relationship
Essential (E)	$\hat{T}R$	$(r^L)^T \Psi^{-T} E \Psi^{-1} r^R = 0$	$(z^L)^T E z^R = 0$
Fundamental (F)	$\Psi^{-T} \hat{T} R \Psi^{-1}$	$(r^L)^T F r^R = 0$	

The difference between the two matrices can be analyzed across four areas: # of parameters, what it maps, solving for the matrix, and using the matrix:

- # Parameters
  - E - 5 parameters (3 for rotation, 2 for translation[18])
  - F - 7 parameters (2 + 2 for epipoles, 3 for homography)
- Solving for the matrix: suppose you have two images taken from stereo cameras of the same object so that there are some corresponding points in both image planes. To solve for F and E, start with their basic formulas and linearize using the method shown in section 2.3.2:[19]

$$0 = (r_i^L)^T F r_i^R \qquad \Rightarrow \qquad 0 = A(r_i^L r_i^R)(\overrightarrow{f})$$

$$0 = (r_i^L)^T \Psi^{-T} E \Psi^{-1} r_i^R \qquad \Rightarrow \qquad 0 = A(r_i^L r_i^R)(\overrightarrow{\psi^{-T} e \psi^{-1}})$$

  - E - requires solving for both extrinsic (R, T) and intrinsic parameters (Ψ)
  - F - requires solving for only extrinsic (R, T) parameters
- Using the matrix: after reviewing the above table, you will note the following:[20]
  - E - requires knowing only extrinsic parameters
  - F - requires knowing both intrinsic and extrinsic parameters
- What it maps - the essential (fundamental) matrix maps rays to rays (points to points) as shown in the below figure

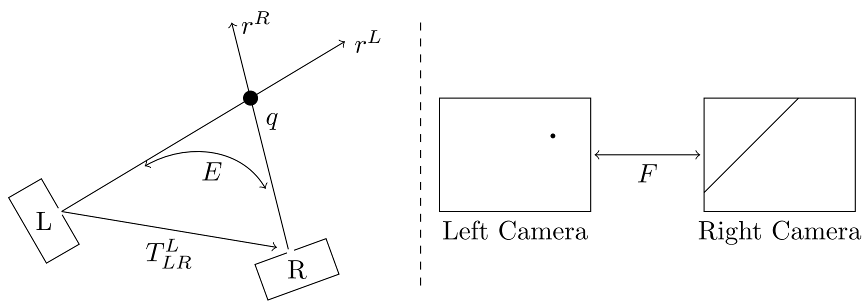
[16] note that $A^{-T} \equiv (A^T)^{-1}$

[17] It turns out that $E = -R^T \hat{T}$ although the sign generally does not matter because it is in an equation set equal to 0 (equation (2). To see this, take the transpose of equation (2) so that $0 = 0^T = (r^R)^T (R^T \hat{T}^T r^L$ where $\hat{T} = \begin{pmatrix} 0 & -T^3 & T^2 \\ T^3 & 0 & -T^1 \\ -T^2 & T^1 & 0 \end{pmatrix} = -\hat{T}^T$ (see equation (1)).

[18] Magnitude does not matter.

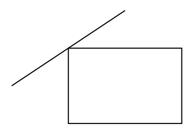
[19] The notation $\overrightarrow{f}$ means "the linearized version of F"

[20] Note: if you want to map r^L to r^R, you will always need Ψ. This bullet is mostly just saying the formulas of E and F are different.

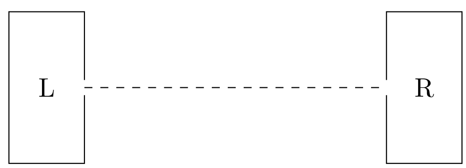


2.5 Areas where epipolar lines fail

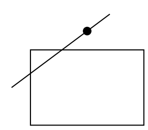
- Epipolar line in one camera does not cross the image plane of the other camera



- Ray from known camera passes through the other camera's optical center



- The point on the epipolar line is outside the field of view



References

[1] P. Vela, "ECE 4580 class lectures," Spring 2013, (Georgia Institute of Technology).

[2] R. Szeliski, *Feature detection and matching.* Springer, 2009.