The purpose of camera calibration is to determine the intrinsic camera parameters \((c_0, r_0, s_x, s_y, f)\), skew parameter \((s = \cot \alpha)\), and the lens distortion (radial distortion coefficient \(k_1\)). Skew parameter defines the angle between the \(c\) and \(r\) axes of a pixel. For most CCD camera, we have rectangular pixel. The skew angle is 90. The skew parameter is therefore zero. \(k_1\) is often very small and can be assumed to be zero. The intrinsic camera matrix is

\[
W = \begin{pmatrix}
    f_{sx} & \cot \alpha & c_0 \\
    0 & \frac{f_{sy}}{\sin \alpha} & r_0 \\
    0 & 0 & 1
\end{pmatrix}
\]

If we assume \(\alpha = 90\), then we have

\[
W = \begin{pmatrix}
    f_{sx} & 0 & c_0 \\
    0 & f_{sy} & r_0 \\
    0 & 0 & 1
\end{pmatrix}
\]

\(f_{sx}\) and \(f_{sy}\) are collectively treated as two separate parameters instead of three parameters.
**Camera Calibration Methods**

- Conventional method: use a calibration pattern that consists of a 2D and 3D data to compute camera parameters. It needs a single image of the calibration pattern.
- Camera self-calibration: only need 2D image data of the calibration pattern but need multiple 2D image data from multiple views. Problem is often less constrained than the conventional method.

---

**Conventional Camera Calibration**

Given a 3D calibration pattern, extract image features from the calibration pattern. Use 2D image features and the corresponding 3D features on the calibration pattern. The features can be points, lines, or curves or their combination.

- Determine the projection matrix $P$
- Derive camera parameters from $P$

$$P_{full} = \begin{pmatrix}
    s_x f r_1^t + c_0 t_3 & s_x f t_x + c_0 t_z \\
    s_y f r_2^t + r_0 t_3 & s_y f t_y + r_0 t_z \\
    r_3^t & t_z
\end{pmatrix}$$

---

**Compute P: Linear Method using 2D/3D Points**

Given image points $m_i^{2 \times 1} = (c_i, r_i)^t$ and the corresponding 3D points $M_i^{3 \times 1} = (x_i, y_i, z_i)^t$, where $i=1, \ldots, N$, we want to compute $P$. Let $P$ be represented as

$$P = \begin{pmatrix}
    p_1^t & p_{14} \\
    p_2^t & p_{24} \\
    p_3^t & p_{34}
\end{pmatrix}$$

where $p_i, i=1,2,3$ are $3 \times 1$ vectors and $p_{14}, i=1,2,3$, are scalers.

---

**Compute P: Linear Method (cont’d)**

Then for each pair of 2D-3D points, we have

$$M_i^t p_1 + p_{14} - c_i M_i^t p_1 - c_i p_{34} = 0$$
$$M_i^t p_2 + p_{24} - r_i M_i^t p_3 - r_i p_{34} = 0$$

For $N$ points, we can setup a system of linear equations

$$AV = 0$$

where $A$ is a $2N \times 12$ matrix depending only on the 3-D and 2-D coordinates of the calibration points, and $V$ is a $12 \times 1$ vector $(p_1^t p_{14} p_2^t p_{24} p_3^t p_{34})^t$. 
\[ A = \begin{pmatrix}
M_1^t & 1 & 0 & -c_1 M_1^t & -c_1 \\
0 & M_1^t & 1 & -r_1 M_1^t & -r_1 \\
& \vdots \\
M_N^t & 1 & 0 & -c_N M_N^t & -c_N \\
0 & M_N^t & 1 & -r_N M_N^t & -r_N 
\end{pmatrix}
\]

where \( \vec{0}^{1 \times 3} = [0 \ 0 \ 0] \)

In general, the rank of \( A \) is 11 (for 12 unknowns), which means the solution is up to a scale factor. But due to effect of noise and locational errors, \( A \) may be full rank, which may make the solution (corresponding to the eigenvector of the smallest eigen value) unique.

The rank of \( A \) may also change for certain special configurations of input 3D points, for example collinear points, coplanar points, etc. The issue is of practical relevance.

For the linear method to work, we need \( N \geq 6 \) and the \( N \) points cannot be coplanar points.

Rank of \( A \) is a function of the input points configurations (see section 3.4.1.3 of Faugeras). If 3D points are coplanar, then rank\( (A) < 11 \) (in fact, it equals 8), which means there is an infinite number of solutions. Faugeras proves that 1) in general Rank\( (A)=11 \); 2) for coplanar points \( (N \geq 4) \), rank\( (A)=8 \) since three points are needed to determine a plane \((11-3=8)\).

How about the rank of \( A \) if points are located a sphere or on the planes that are orthogonal or parallel to each other? Hint: how many points are needed to determine a sphere?
**Linear Solution 1**

Minimize

$$||AV||^2$$

to solve for $V$. Can we solve for $V$? Solution to $AV = 0$ is not unique (up to a scale factor). It lies in the null space of $A$. If rank($A$) = 11, then $V$ is the null vector of $A$, multiplied by a scalar. If on the other hand, rank($A$) $\leq$ 11, the the solution to $X$ is the linear combinations of all null vectors of $A$.

If rank($A$) = 11, then $V$ is the null vector of $A$, multiplied by a scaler. The null vector $A$ can be obtained by performing SVD on $A$, yielding

$$A_{m \times n} = U_{m \times m} D_{m \times n} (S_{n \times n})$$

The null vector $A$ is the last column of $S$ matrix. Note $V$ is solved only up to a scale factor. The scale factor can be recovered using the fact that $||p_3||^2 = 1$.


Alternatively, we can also solve $V$ by minimizing $||AV||^2$, which yields $(A'AV = 0) \text{ or } (A'AV = \lambda V)$, where $\lambda = 0$. As a result, solution to $V$ is the eigen vector of matrix $(A'AV)$ corresponding to zero eigen value. This implies that the eigenvectors of $A'AV$ correspond to the columns of the $S$ matrix.

**Linear Solution 2**

Let $A = [B \ b]$, where

$$B = \begin{pmatrix}
M_1^t & 1 & 0 & -c_1M_1^t \\
0 & 0 & M_1^t & 1 & -r_1M_1^t \\
\vdots & \vdots & \vdots & \vdots \\
M_N^t & 1 & 0 & -c_NM_N^t \\
0 & 0 & M_N^t & 1 & -r_NM_N^t
\end{pmatrix}$$

$$b = (-c_1 - r_1 \ldots - c_N - r_N)^t$$

Then, $AV = p_{34}(BY + b)$. Since $p_{34}$ is a constant, minimizing $||AV||^2$ is the same as minimizing $||BY + b||^2$, whose solution is $Y = -(B'B)^{-1}B'b$. The rank of matrix $B$ must be eleven.
**Linear Solution 2 (cont’d)**

The solution to $Y$ is up to a scale factor $p_{34}$. To recover the scale factor, we can use the fact that $|p_{34}| = 1$. The scale factor $p_{34}$ can be recovered as $p_{34} = \frac{1}{\sqrt{Y^2(9)+Y^2(10)+Y^2(11)}}$, where $Y(9)$, $Y(10)$, and $Y(11)$ are the last 3 elements of $Y$. The final projection matrix (vector) is therefore equal to

$$V = \begin{pmatrix} p_{34}Y \\ p_{34} \end{pmatrix}$$

**Linear Solution 3**

Imposing the orthonormal constraint, $R^t = R^{-1}$, i.e., minimize $||AV||^2$ subject to $R^t = R^{-1}$. Solution to this problem is non-linear !.

**Linear Solution 3 (cont’d)**

To yield a linear solution, we can impose one of the normal constraints, i.e., $||p_{34}||^2 = 1$, then the problem is converted to a constrained linear least-squares problem. That is, minimize $||AV||^2$ subject to $||p_{34}||^2 = 1$.

$$e^2 = ||AV||^2 + \lambda(||p_{34}||^2 - 1)$$

(2)

Decomposing $A$ into two matrices $B$ and $C$, and $V$ into $Y$ and $Z$

$$A = \begin{pmatrix} B & C \end{pmatrix}$$

$$V = \begin{pmatrix} Y \\ Z \end{pmatrix}$$

$$B^{2N \times 9} = \begin{pmatrix} M^t_1 & 1 & 0 & -c_1 \\ \delta & 0 & M^t_1 & 1 & -r_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ M^t_N & 1 & \delta & 0 & -c_N \\ \delta & 0 & M^t_N & 1 & -r_N \end{pmatrix}$$

$$C^{2N \times 3} = \begin{pmatrix} -c_1 M^t_1 \\ -r_1 M^t_1 \\ \vdots \\ -c_N M^t_N \\ -r_N M^t_N \end{pmatrix}$$
\[ Y = \begin{pmatrix} p_1 \\ p_{14} \\ p_2 \\ p_{24} \\ p_{34} \end{pmatrix} \quad Z = p_3 \]

Then equation 2 can be rewritten as
\[
\epsilon^2 = ||BY + CZ||^2 + \lambda(||Z||^2 - 1)
\]

Taking partial derivatives of \( \epsilon^2 \) with respect to \( Y \) and \( Z \) and setting them to zeros yield
\[
Y = -(B^tB)^{-1}B^tCZ \\
C^t(I - B(B^tB)^{-1}B^t)CZ = \lambda Z
\]

Apparently, the solution to \( Z \) is the eigenvector of matrix \( C^t(I - B(B^tB)^{-1}B^t)C \). Given \( Z \), we can then obtain solution to \( Y \).

\[
\text{Substituting } Y \text{ into } ||BY + CZ||^2 \text{ leads to}
\]
\[
||BY + CZ||^2 = || - B(B^tB)^{-1}B^tCZ + CZ||^2 \\
= ||(I - B(B^tB)^{-1}B^t)CZ||^2 \\
= Z^tC^t(I - B(B^tB)^{-1}B^t)(I - B(B^tB)^{-1}B^t)CZ \\
= Z^tC^t(I - B(B^tB)^{-1}B^t)CZ \\
= Z^t\lambda Z \\
= \lambda
\]

This proves that solution to \( Z \) corresponds to the eigenvector of the smallest positive eigenvalue of matrix \( C^t(I - B(B^tB)^{-1}B^t)C \).

Note \((I - B(B^tB)^{-1}B^t)(I - B(B^tB)^{-1}B^t) = (I - B(B^tB)^{-1}B^t)\)
Linear Calibration with Planar Object

Since planar points reduces the rank of A matrix to 8, we cannot follow the conventional way of camera calibration using planar object. But this becomes possible if we acquire two images of the planar object, producing two A matrices. With each A providing 8 independent equations, we can have a total of 16 independent equations, theoretically sufficient to solve for the intrinsic matrix W. But since the two A matrices share the same W but different extrinsic parameters, the extrinsic parameters must be eliminated from the system of linear equations to only determine W. This can be done using the homography equation. For details see Zhengyou Zhang’s calibration method at http://research.microsoft.com/en-us/um/people/zhang/Calib/

Other Linear Techniques


Compute P: Non-linear Method

Let the 3D points be $M_i = (x_i, y_i, z_i)^T$ and the corresponding image points be $m_i = (c_i, r_i)^T$ for $i = 1, 2, \ldots, N$. The criterion function to minimize is

$$
\epsilon^2 = \sum_{i=1}^{N} \left( \frac{M_i^T p_1 + p_{14}}{M_i^T p_3 + p_{34}} - c_i \right)^2 + \left( \frac{M_i^T p_2 + p_{24}}{M_i^T p_3 + p_{34}} - r_i \right)^2
$$

subject to $||p_3||^2 = 1$ and $(p_1 \times p_3) \cdot (p_2 \times p_3) = 0$, i.e., the orthonormal constraints on $R$.

Introducing the lagrangian multipliers, we have the constrained objective function

$$
\epsilon^2 = \sum_{i=1}^{N} \left( \frac{M_i^T p_1 + p_{14}}{M_i^T p_3 + p_{34}} - c_i \right)^2 + \left( \frac{M_i^T p_2 + p_{24}}{M_i^T p_3 + p_{34}} - r_i \right)^2 + \lambda_1 (||p_3||^2 - 1) + \lambda_2 [(p_1 \times p_3) \cdot (p_2 \times p_3)]
$$

First order gradient descent may be used to solve $P$ iteratively. For example, we can solve $p_1$ as follows

$$
p_1 = p_1^{t-1} - \alpha \nabla_{p_1} \epsilon^2
$$

where $\alpha$ is the learning rate and $\nabla_{p_1} \epsilon^2$ is the gradient for $p_1$. It can be
computed as follows

$$\nabla_{\vec{p}_1} \epsilon^2 = \sum_{i=1}^{N} 2(\frac{M_i^T \vec{p}_1 + p_{14}}{M_i^T \vec{p}_3 + p_{34}} - c_i)(\frac{M_i}{M_i^T \vec{p}_3 + p_{34}})$$

$$+ \lambda_2 \frac{\partial(p_1 \times p_3)}{\partial p_1}(p_2 \times p_3)$$

where

$$\frac{\partial(p_1 \times p_3)}{\partial p_1} = p_1 \times \frac{\partial p_3}{\partial p_1} + p_3 \times \frac{\partial p_1}{\partial p_1}$$

$$= p_3 \times \frac{\partial p_1}{\partial p_1}$$

Note other criterion function such as the Sampson error (section 3.2.6 of Hartley’s book) function. Sampson represents the first order approximation to the geometric error in equation 3.

Another way of solving this problem is to perform minimization directly with respect to the intrinsic and extrinsic parameters.

Linear Method v.s Non-linear Method

- Linear method is simple but less accurate and less robust
- Linear solution can be made robust via the robust method such as the RANSAC method
- Linear method does not require initial estimate
- Non-linear method is more accurate and robust but complex and require good initial estimates

The common approach in CV is two steps:

- Use a linear method to obtain initial estimates of the camera parameters.
- Refine the initial estimates using an non-linear method.
Data Normalization

Hartley introduces a data normalization technique to improve estimation accuracy. Details of the normalization technique may be found on section 3.4.4 of Hartley's book. This normalization should precede all estimation that involves image data.

A brief discussion of this normalization procedure can also be found at page 156 of Trucco's book.

Robust Linear Method with RANSAC

The linear LSQ method is sensitive to image errors and outliers. One solution is to use a robust method. The most commonly used robust method in CV is the RANSAC (Random Sample Consensus) method. It works as follows:

- Step 1: Randomly pick a subset of K points from N (K > 6) pixels in the image and compute the projection matrix $P$ using the selected points.
- Step 2: For each of the remaining pixels in the image, compute its projection error using the $P$ computed from step 1. If it is within a threshold distance, increment a counter of the number of points (the "inliers") that agree with the hypothesized $P$.
- Step 3: Repeat Steps 1 and 2 for a sufficient number of times $a$, and then select the subset of points corresponding to the $P$ with the largest count. The exact number of times is determined by the required probability that one of subset does not contain the outliers.

Compute Camera Parameters from $P$

$$P = \begin{pmatrix} s_x f r_1^t + c_0 r_3^t & s_x f t_x + c_0 t_z \\ s_y f r_2^t + r_0 r_3^t & s_y f t_y + r_0 t_z \\ r_3 & t_z \end{pmatrix}$$

- $r_3 = p_3$
- $t_z = p_{34}$
- $c_0 = p_1^t p_3$
- $r_0 = p_2^t p_3$
- $s_x f = \sqrt{p_1^t p_1 - c_0^2} = ||p_1 \times p_3||$
- $s_y f = \sqrt{p_2^t p_2 - r_0^2} = ||p_2 \times p_3||$
- $t_x = (p_{14} - c_0 t_z)/(s_x f)$
- $t_y = (p_{24} - r_0 t_z)/(s_y f)$
Compute Camera Parameters from $P$ (cont’d)

Alternatively, we can compute $W$ algebraically from $P$. Since $P = WM = W[RT] = [WR WT]$, let $P_3$ be the first $3 \times 3$ submatrix of $P$, the $P_3 = WR$.

Hence,

$$K = P_3^T P_3 = WW^t$$

Given $K$, from equation $K = WW^t$, we can obtain $W$ via Choleski factorization. Since $K$ is symmetric, from SVD, we have:

$$K = UDU^t$$

where $D$ is a diagonal matrix and $U$ an orthonormal matrix, whose columns are the eigenvectors of $K$. Since $D$ is diagonal, we may write $D = D^{1/2}D^{1/2}$.

As a result,

$$K = HH^t$$

where $H = U D^{1/2}$. Since $H$ is not upper triangular yet, we can perform a RQ decomposition on $H$, this yields to $H = BQ$, where $B$ is an upper triangular matrix and $Q$ is an orth-normal matrix. $W$ equals to $B$.

Given $W$, $T$ can be recovered as $T = W^{-1}P_4$, where $P_4$ is the last column of $P$, and $R = W^{-1}P_3$.

Approximate solution to imposing orthonormality

Let $\hat{R}$ be the estimated rotation matrix $R$. It is not orthonormal. We can find an orthonormal matrix $\tilde{R}$ that is closest to $\hat{R}$ via SVD. Let $\hat{R} = UDV^t$, find another matrix $E$ that is closest to $D$ and that satisfies $E^{-1} = E^t$. If we take $E$ be a diagonal matrix, then we have $E = I$, the identity matrix. As a result, we have a new estimate of $R$, which is $\tilde{R} = UIV^t$. $\tilde{R}$ is the orthonormal matrix that is closest to $\hat{R}$.
Image Center using Vanishing Points

Let \( L_i, i=1,2, \ldots, N \) be parallel lines in 3D, \( l_i \) be the corresponding image lines. Due to perspective projection, lines \( l_i \) appear to meet in a point, called vanishing point, defined as the common intersection of all the image lines \( l_i \).

Given the orientation of the \( L_i \) lines be \( N = (n_x, n_y, n_z)^t \) relative to the camera frame, then the coordinates of the vanishing point in the image frame are \( (\frac{n_x}{n_z}, \frac{n_y}{n_z}) \).

Let \( T \) be the triangle on the image plane defined by the three vanishing points of three mutually orthogonal sets of parallel lines in space. The image center, i.e., the principal point \( (c_0, r_0) \), is the orthocenter \(^a \) of \( T \).

\(^a\)it is defined as the intersections of the three altitudes.

Non-Linear Direct Solution to Camera Parameters

Let

\[
\Theta = (c_0 \ v_0 f \ t_x \ t_y \ t_z)^t
\]

\[
g(\Theta, M_i) = \frac{M_i^t p_i + p_i}{M_i^t p_i + p_i^t}
\]

\[
f(\Theta, M_i) = \frac{M_i^t p_i^t + p_i}{M_i^t p_i + p_i^t}
\]

Then the problem can be stated as follows:

Find \( \Theta \) by minimizing

\[
\epsilon^2 = \sum_{i=1}^{N} [g(M_i, \Theta) - c_i]^2 + [f(M_i, \Theta) - r_i]^2
\]

Gradient descent can be used to solve for each parameter iteratively, i.e.,

\[
\Theta^t = \Theta^{t-1} - \alpha \nabla \epsilon^2
\]

Other methods to solve for non-linear optimization include Newton method, Gauss-Newton, and Levenberg-Marquardt method. See chapter 3 of Forsyth and Ponce’s book on how these methods work. Refer to appendix 4 of Hartley’s book for additional iterative estimation methods. Non-linear methods all need good initial estimates to correctly converge. Implement one of the non-linear method using Matlab. It could improve the results a lot.

Camera Calibration using Lines and Conics

Besides using points, we can also perform camera calibration using correspondences between 2D/3D lines and 2D/3D conics. Furthermore, we can also extend the point-based camera calibration to estimate the lens distortion coefficient \( k_1 \). These can be topics for the final project.
Camera Calibration under Weak Perspective Projection

For weak perspective projection, we have
\[
\begin{pmatrix}
\hat{c} \\
\hat{r}
\end{pmatrix} = M_{2\times3} \begin{pmatrix}
\bar{x} \\
\bar{y}
\end{pmatrix}
\]

Given 2D/3D relative coordinates \((\hat{c}_i, \hat{r}_i)\) and \((\bar{x}_i, \bar{y}_i, \bar{z}_i)\), the goal is to solve for matrix \(M\). A minimum of 3 points are enough to uniquely solve for the matrix \(M\). And, more importantly, these points can be coplanar points.

Camera Calibration with Weak Perspective Projection

Given \(M\) and the parameterization for \(M\) introduced in the previous chapter, we have
\[
\begin{align*}
\frac{f_x}{\bar{z}_c} &= |m_1| \\
\frac{f_y}{\bar{z}_c} &= |m_2|
\end{align*}
\]
where \(m_1\) and \(m_2\) are the first row and the second row of the \(M\) matrix.

Then,
\[
\begin{align*}
r_1 &= \frac{m_1}{|m_1|} \\
r_2 &= \frac{m_2}{|m_2|} \\
r_3 &= r_1 \times r_2
\end{align*}
\]

Camera Calibration with Lens Distortion

We present an approach for simultaneous linear estimation of the camera parameters and the lens distortion, based on the division lens distortion model proposed by Fitzgibbon\(^a\). According to the divisional model, we have
\[
\begin{pmatrix}
\hat{c} - c_0 \\
\hat{r} - r_0
\end{pmatrix} = (1 + ks^2) \begin{pmatrix}
c - c_0 \\
r - r_0
\end{pmatrix}
\]
where \(s^2 = (\hat{c} - c_0)^2 + (\hat{r} - r_0)^2\). This is an approximation to the camera
\(^a\)The paper appears in CVPR 01, pages 125-132
true distortion model. Hence,

\[ \frac{\lambda}{1 + ks^2} \begin{pmatrix} \hat{c} - c_0 \\ \hat{r} - r_0 \\ 0 \end{pmatrix} = P \begin{pmatrix} x \\ y \\ z \end{pmatrix} - \lambda \begin{pmatrix} c_0 \\ r_0 \\ 1 \end{pmatrix} \]

After solving for \( \lambda = p_3(x y z 1)^t \) and with some algebraic simplifications yield

\[ (D_1 + kD_2)V = 0 \]

where \( V=(p_1 p_2 p_3)^t \), \( p_i \) is the ith row of matrix \( P \), and

\[
D_1 = \begin{pmatrix} x & y & z & 1 & 0 & 0 & 0 & 0 & -\hat{c}x & -\hat{c}y & -\hat{c}z & -c_0 \\ 0 & 0 & 0 & 0 & x & y & z & 1 & -\hat{r}x & -\hat{r}y & -\hat{r}z & -r_0 \end{pmatrix}
\]

Alternatively, we can perform alternation, i.e., assume image center as the principal point, then use the above to compute \( k \) and the internal camera parameters. Then, substitute the computed center back to recompute \( k \) and the camera parameters. This process repeats until it converges, i.e., when the change in the estimated parameters is small.

The procedure can be summarized as follows:

1. Assume principal center is at image center. This allows to compute \( \hat{c} - c_0, \hat{r} - r_0 \), and \( s^2 = (\hat{c} - c_0)^2 + (\hat{r} - r_0)^2 \) for each point.
2. Use polyeig to solve for \( k \) and matrix \( P \)
3. Obtain the intrinsic camera parameters from \( P \)
4. Repeat steps 2) and 3) with the new principal center until the change in the computed intrinsic parameters is less than a pre-defined threshold.

Degeneracy with Camera Calibration

Degeneracy occurs when the solution to the projection matrix is not unique due to special spatial point configurations.

see sections 3.2.3 and 3.3.3 of Forsyth’s book.

MATLAB function \texttt{polyeig} can be used to obtain the solution for both \( k \) and \( V \).

To use \texttt{polyeig} function, the matrices on the left side of above equations must be square matrices. To achieve this, multiple both sides of the above equation by \((D_1 + kD_2)^t\) yields the following, which can be solved for using \texttt{polyeig}

\[
(D_1^t D_1 + k(D_1^t D_2 + D_2^t D_1) + k^2 D_1^t D_2) V = 0^b
\]

The solution, however, assumes the knowledge of the image center. We can fix it as the center of the image. Study shows that the precise location of the distortion center does not strongly affect the correction (see Ref. 9 of Fitz’s paper).

\(^b\)when \( k \) is small, \( D_1 \) is close to the \( A \) matrix.
**Camera Self Calibration**

Self camera calibration refers to determining the interior camera parameters of a camera by using only image data obtained from different view directions or different view points.

Either camera or the object must move to acquire different images.

---

**Methods for Camera Self-Calibration**

- General camera movement (involving both rotation and translation)
- Only rotational movement (same view point but different view directions)
- Only translational movement (different view points but same view direction)

---

**Camera Self-Calibration With Only Rotation**

By exploiting the fact that we do not have 3D information, we can locate the object frame anywhere we want to simplify the subsequent calculations.

Let's assume that we select the initial camera frame as the reference frame and the object frame coincide with the initial camera frame. Let the image generated by the initial camera frame be represented with subscript 0.
\[
\lambda_0 \begin{pmatrix} c_0 \\ r_0 \\ 1 \end{pmatrix} = WM_0 \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = W \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = W \begin{pmatrix} 1 \\ 0 \end{pmatrix}
\]

If we rotate the camera frame from the reference by a rotation matrix \( R_i \), we have
\[
\lambda_i \begin{pmatrix} c_i \\ r_i \\ 1 \end{pmatrix} = WM_i \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = WR_i \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}.
\]

This leads to three equations
\[
c_i = \lambda_i (B_{11}c_0 + B_{12}r_0 + B_{13}) \\
r_i = \lambda_i (B_{21}c_0 + B_{22}r_0 + B_{23}) \\
1 = \lambda_i (B_{31}c_0 + B_{32}r_0 + B_{33})
\]

Since \( \lambda_i = 1/(B_{31}c_0 + B_{32}r_0 + B_{33}) \), substituting \( \lambda_i \) to the above equations yields
\[
c_i(B_{31}c_0 + B_{32}r_0 + B_{33}) = (B_{11}c_0 + B_{12}r_0 + B_{13}) \\
r_i(B_{31}c_0 + B_{32}r_0 + B_{33}) = (B_{21}c_0 + B_{22}r_0 + B_{23})
\]

Given \( N \) points, we can set up a system of linear equations, through which we can solve \( B \) as the null vector of the measurement matrix up to a scale factor.

Alternatively, we can divide the both sides of the above equations by \( B_{33} \) and

Denote \( \lambda_i = \frac{1}{B_{33}} \), substituting 5 to 6 to remove \((X, Y, Z)^T\) yields
\[
\begin{pmatrix} c_i \\ r_i \\ 1 \end{pmatrix} = \lambda_i WR_i W^{-1} \begin{pmatrix} c_0 \\ r_0 \\ 1 \end{pmatrix}
\]

Let \( B_i = WR_i W^{-1} = \begin{pmatrix} B_{i11} & B_{i12} & B_{i13} \\ B_{i21} & B_{i22} & B_{i23} \\ B_{i31} & B_{i32} & B_{i33} \end{pmatrix} \), we have
\[
\begin{pmatrix} c_i \\ r_i \\ 1 \end{pmatrix} = \lambda_i \begin{pmatrix} B_{i11} & B_{i12} & B_{i13} \\ B_{i21} & B_{i22} & B_{i23} \\ B_{i31} & B_{i32} & B_{i33} \end{pmatrix} \begin{pmatrix} c_0 \\ r_0 \\ 1 \end{pmatrix}
\]

they can be rewritten in matrix format
\[
\begin{pmatrix} -c_0 & -r_0 & -1 \\ 0 & 0 & c_i c_0 + c_i r_0 \\ 0 & 0 & -c_i - r_0 \end{pmatrix} \begin{pmatrix} \mathbf{b}_1 \end{pmatrix} = \begin{pmatrix} -c_i \\ -r_i \end{pmatrix}
\]

where \( \mathbf{b}_1 = (B_{i11} B_{i12} B_{i13} B_{i21} B_{i22} B_{i23} B_{i31} B_{i32} B_{i33})^T / B_{i33} \). If we know at least 4 points in two images (such as \( i = 0, 1 \)), we can solve for \( \mathbf{b}_1 \) up to a scale factor. The scale factor can be solved using the fact that the determinant of \( WR_i W^{-1} \) is unit.

If \( R_i \) is known, then we can solve for \( W \) using the equation \( B_i W = WR_i \). In this case, one rotation, i.e., a total of two images, is enough to solve for \( W \).

To solve for \( W \) with an unknown \( R_i \). From \( \mathbf{b}_1 = WR_i W^{-1} \), we have
\[
R_i = W^{-1} B_i W \quad R_i^{-T} = W^T B_i^{-1} W^{-T}
\]
Since $R = R^{-T}$, therefore we have

$$(WW^T)B_i^{-T} = Bi(WW^T)$$

Assume $C = WW^T$, we have

$$C = \begin{pmatrix} s_x f & 0 & c_0 \\ 0 & s_y f & r_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} s_x f & 0 & 0 \\ 0 & s_y f & 0 \\ c_0 & r_0 & 1 \end{pmatrix} = \begin{pmatrix} s_x^2 f^2 + c_0^2 & c_0 r_0 & c_0 \\ c_0 r_0 & s_y^2 f^2 + r_0^2 & r_0 \\ c_0 & r_0 & 1 \end{pmatrix}$$

\[ (12) \]

Equation 11 can be rewritten

$$B_iCB_i^T - C = 0$$

Since $C$ is symmetric, Eq. 13 provides only six equations. To solve for $C$, it is necessary to use two $B_i$, i.e., two rotations, which leads to three images. Given two or more $B_i$, $C$ can be solved using equation 13 up to a scale factor. The scale factor can subsequently be resolved using the fact that the last element of $W$ is 1.

Given $C$, from equation $C = WW^T$, we can obtain $W$ via SVD and RQ decomposition as detailed below. Since $C$ is symmetric, from SVD, we have:

$$C = UDU^T$$

where $D$ is a diagonal matrix and $U$ an orthonormal matrix, whose columns are the eigenvectors of $C$. Since $D$ is diagonal, we may write $D = EE^t$, where $E$ is another diagonal matrix. As a result,

$$C = VV^t$$

where $V = UE$. Note the above SVD can be directly obtained via Choleski factorization. Since $V$ is not upper triangular yet, we can perform a RQ decomposition on $V$, this yields to $V = BQ$, where $B$ is an upper triangular matrix and $Q$ is just an orthonormal matrix. Hence, $W = B$. We can also prove that the solution is unique. The solution requires $C$ be positive definite.

Camera Self-Calibration With Only Translation

Like for the previous case, the camera frame coincides with the object frame. We then translate the camera frame by $T_i$. For the image points in the reference frame and the newly translated frame, we have

$$\lambda_0 \begin{pmatrix} c_0 \\ r_0 \\ 1 \end{pmatrix} = W[I T_0] \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = W \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

$$\lambda_i \begin{pmatrix} c_i \\ r_i \\ 1 \end{pmatrix} = W[I T_i] \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = W \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + WT_i$$

\[ (14) \] \[ (15) \]
From the above equations, we have
\[
\lambda_i \begin{pmatrix} c_i \\ r_i \\ 1 \end{pmatrix} = \lambda_0 \begin{pmatrix} c_0 \\ r_0 \\ 1 \end{pmatrix} + \mathbf{W} \mathbf{T}_i \tag{16}
\]

Equation 16 can be rewritten
\[
\lambda_i \begin{pmatrix} c_i \\ r_i \\ 1 \end{pmatrix} = \lambda_0 \begin{pmatrix} c_0 \\ r_0 \\ 1 \end{pmatrix} + \begin{pmatrix} s_x f t_i_x + c_0 t_i_z \\ s_y f t_i_y + r_0 t_i_z \\ t_i_x \\ t_i_y \end{pmatrix} \tag{17}
\]

then, we get three equations, assuming \( T \) is known
\[
\lambda_i c_i = \lambda_0 c_0 + (s_x f t_i_x + c_0 t_i_z) \\
\lambda_i r_i = \lambda_0 r_0 + (s_y f t_i_y + r_0 t_i_z)
\]

(There is no linear solution if \( T \) is unknown)

\[
\lambda_i = \lambda_0 + t_i_z \tag{18}
\]

Substituting \( \lambda_i = \lambda_0 + t_i_z \) to the above two equations yields
\[
\lambda_0 c_i + t_i_z c_i = \lambda_0 c_i + s_x f t_i_x + c_0 t_i_z \\
\lambda_0 r_i + t_i_z r_i = \lambda_0 r_i + s_y f t_i_y + r_0 t_i_z \tag{19}
\]

Equation 16 can be rewritten
\[
\begin{pmatrix} t_i_x \\ 0 \\ t_i_y \\ 0 \end{pmatrix} = \begin{pmatrix} t_i_x c_i \\ 0 \\ t_i_y r_i \\ 0 \end{pmatrix} \tag{20}
\]

where \( \mathbf{W}' = (s_x f \ s_y f \ c_0 \ r_0 \ \lambda_0)^T \). If we know at least 3 points in three images (produced by two translations), we can solve \( \mathbf{W}' \), then we can get the solution to \( \mathbf{W} \). Note the matrix is rank deficient (rank=3) (since the first 2x4 sub-matrix is the same for all points) if only using one translation, no matter how many points are used.

**Camera Self Calibration Summary**

Camera self calibration can be carried out without using 3D data. It requires camera to move to produce different images. Camera motions can be

- general motion-involving both rotation and translation. Solution is unstable and less robust
- pure rotation-requires a minimum of two rotations (or three images) if rotation is unknown. If rotation is known, one rotation or two images is enough.
- pure translation-requires a minimum of two translations and they must be known to have a linear solution.
- degenerate motions may happen and they cannot be used for self calibration.

**Pose Estimation**

The goal of pose estimation is to estimate the relative orientation and position between the object frame and the camera frame, i.e., determining \( R \) and \( T \) or extrinsic camera parameters.

Pose estimation is an area that has many applications including HCI, robotics, photogrametry, etc..
Linear Pose Estimation

For pose estimation, it is necessary to know 3D features and their 2D image projections. They are then used to solve for R and T.

Assume W is known, then from the projection equation

\[ \lambda W^{-1} \begin{pmatrix} c \\ r \\ 1 \end{pmatrix} = [R, T] \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \]

Given more than 6 sets of 2D/3D points, we can solve for R and T linearly in the similar fashion to that of linear camera calibration. The solution, however, does not impose the constraint that R be orthnormal, i.e., \( R^{-1} = R^t \).

We can perform a postprocessing of the estimated R to find another \( \hat{R} \), that is closest to R and orthnormal. See previous pages for details.

Alternatively, we can still have a linear solution if we impose one constraint, i.e., \( ||r_3|| = 1 \) during optimization.

If W is unknown, we can follow the same procedure as camera calibration to first solve for P and then extract R and T from P.

Non-linear Pose Estimation (cont'd)

Alternatively, we can set it up as a non-linear optimization problem, with the constraint of \( R^{-1} = R^t \) imposed.

Pose Estimation Under Weak Perspective Projection

For weak perspective projection, we have

\[ \begin{pmatrix} \bar{c} \\ \bar{r} \end{pmatrix} = M_{2 \times 3} \begin{pmatrix} \bar{x} \\ \bar{y} \\ \bar{z} \end{pmatrix} \]

Given 2D/3D relative coordinates \((\bar{c}_i, \bar{r}_i)\) and \((\bar{x}_i, \bar{y}_i, \bar{z}_i)\), the goal is to solve for matrix \( \tilde{M} \). A minimum of 3 points are enough to uniquely solve for the matrix \( \tilde{M} \).
Given \( M \) and the parameterization for \( M \) introduced in the previous chapter, we have
\[
\frac{f s_x}{z_c} = |m_1| \\
\frac{f s_y}{z_c} = |m_2| 
\]
where \( m_1 \) and \( m_2 \) are the first row and the second row of the \( M \) matrix.

Then,
\[
\begin{align*}
    r_1 &= \frac{m_1}{|m_1|} \\
    r_2 &= \frac{m_2}{|m_2|} \\
    r_3 &= r_1 \times r_2
\end{align*}
\]

If \( W \) is unknown,
\[
\left( \begin{array}{c} c \\ r \\ 1 \end{array} \right) = M_{2\times3} \left( \begin{array}{c} x \\ y \\ z \end{array} \right) + \left( \begin{array}{c} v_x \\ v_y \end{array} \right)
\]
where \( v_x = \frac{p_{14}}{p_{34}} = \frac{f s_x t_x + c_0}{z_c} \) and \( v_y = \frac{p_{24}}{p_{34}} = \frac{f s_y t_y + r_0}{z_c} \). If we assume \( c_0 \) and \( r_0 \) are in image center, we can solve for the translation \( t_x \) and \( t_y \) up to a scale factor.