Camera Calibration and Pose Estimation

3D point in object frame

- 3D reconstruction, motion estimation, object tracking, etc..
  - Affine transformation
    - 3D point in camera frame
      - Perspective projection
        - Image point in image frame
          - Spatial Sampling
            - Image point in row-column frame

\[ M = [R' T] \]

- Pose Estimation
- W
  - Camera Calibration
Camera Calibration and Pose Estimation

The purpose of camera calibration is to determine intrinsic camera parameters: $c_0, r_0, s_x, s_y,$ and $f$. Camera calibration is also referred to as interior orientation problem in photogrammetry.

The goal of pose estimation is to determine exterior camera parameters: $\omega, \phi, \kappa, t_x, t_y,$ and $t_z$. In other words, pose estimation is to determine the position and orientation of the object coordinate frame relative to the camera coordinate frame or vice versa.
Camera Calibration

The purpose of camera calibration is to determine the intrinsic camera parameters \((c_0, r_0), f, s_x, s_y\), skew parameter \((s = \cos \alpha)\), and the lens distortion (radial distortion coefficient \(k_1\)). Skew parameter is a function of the skew 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

\[
W = \begin{pmatrix}
  f s_x & \cos \alpha & c_0 \\
  0 & \frac{f s_y}{\sin \alpha} & r_0 \\
  0 & 0 & 1
\end{pmatrix}
\]

If we assume \(\alpha = 90\), then we have
\[ W = \begin{pmatrix} f s_x & 0 & c_0 \\ 0 & f s_y & r_0 \\ 0 & 0 & 1 \end{pmatrix} \]

\( f s_x \) and \( f s_y \) 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 $P$

\[
P_{full} = \begin{pmatrix}
    s_x f r_1 + c_0 r_3 & s_x f t_x + c_0 t_z \\
    s_y f r_2 + r_0 r_3 & s_y f t_y + r_0 t_z \\
    r_3 & t_z
\end{pmatrix}
\]
Linear Algebra Basics

Understand the followings

- Range, span, null space, determinant, and rank of a matrix
- Eigen-value and eigen-vector of a matrix
- Pseudo-inverse (generalized) inverse of a matrix
- Singular value decomposition (SVD) and its relationship to eigen-value and eigen-vector
- Setup of a system of linear equations and its linear least squares solution
- Homogeneous coordinate system
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

\[
P = \begin{pmatrix}
  p_1^T & p_{14} \\
p_2^T & p_{24} \\
p_3^T & p_{34}
\end{pmatrix}
\] (1)

where \( p_i, i = 1, 2, 3 \) are \( 3 \times 1 \) vectors and \( p_{i4}, i = 1, 2, 3 \), are scalers.
Then for each pair of 2D-3D points, we have

\[
M_i^T p_1 + p_{14} - c_i M_i^T p_3 - c_i p_{34} = 0
\]

\[
M_i^T p_2 + p_{24} - r_i M_i^T p_3 - r_i p_{34} = 0
\]

Each pair of 2D/3D points gives two equations on the projection matrix. For 12 unknowns in the projection matrix, we need a minimum of \(N=6\) pairs of 2D/3D points. If \(N < 6\), there is an infinite number of solutions. For \(N = 6\) points, depending on the condition of \(A\) matrix, there can be one unique exact solution. We can setup a system of linear equations to exactly solve for \(V\), i.e.,

\[
AV = 0
\]

where \(A\) is a \(12 \times 12\) matrix depending only on the 3-D and 2-D
coordinates of the calibration points, and \( V \) is a \( 12 \times 1 \) vector 
\[
(p_1^T \ p_{14}^T \ p_2^T \ p_{24}^T \ p_3^T \ p_{34})^T.
\]
Solutions to the linear systems of equations include direct methods (e.g. variable elimination, Gaussian elimination) or the iterative conjugate gradient method for large sparse linear systems.
\[ A = \begin{pmatrix}
M_1^T & 1 & \vec{0} & 0 & -c_1 M_1^T & -c_1 \\
\vec{0} & 0 & M_1^T & 1 & -r_1 M_1^T & -r_1 \\
\vdots & & & & \\
M_N^T & 1 & \vec{0} & 0 & -c_N M_N^T & -c_N \\
\vec{0} & 0 & M_N^T & 1 & -r_N M_N^T & -r_N \\
\end{pmatrix} \]

where \( \vec{0}^{1 \times 3} = [0 \ 0 \ 0] \)
To have a robust and accurate solution, we choose $N$ such that $N > 6$. In this case, there is not an exact solution; but we can find one unique linear least squares solution (depending on the condition of A matrix) that satisfies the linear equations as closely as possible, by minimizing $\epsilon^2 = ||AV||_2^2$, i.e.,

$$V^* = \arg \min_V ||AV||_2^2$$

Taking derivative of $\epsilon^2 = ||AV||_2^2$ w.r.t $V$ and setting it to zero yields

$$\frac{\partial \epsilon^2}{\partial V} = \frac{\partial (AV)^T (AV)}{\partial V} = 0$$

$$A^T AV = 0$$
For the linear method to produce a unique solution, we need $N \geq 6$ and the $N$ points cannot be coplanar points.
The solutions depend on the rank of A. In general, given \( AV = 0 \), the rank of A is less than 12, which means the solution is not unique. But due to effect of noise and locational errors, A may be full rank, which means only the trivial solution \( V = 0 \) exists.

Besides the number of points, 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.
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 $\text{rank}(A) < 11$ (in fact, it equals 8), which means there is an infinite number of solutions. Faugeras proves that 1) in general for non-coplanar of more than 6 points, $\text{Rank}(A)=11$; 2) for coplanar points ($N \geq 4$), $\text{rank}(A)=8$ 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?
Given $\epsilon^2 = \|AV\|_2^2$, the linear least-squares solution is

$$V^* = \arg \min_V \epsilon^2,$$

which leads to

$$A^T AV = 0$$

Solution to $A^T AV = 0$ is not unique (up to a scale factor). It lies in the null space of $A^T A$. As the null space of $A^T A$ is the same as that of $A$, we can find the null space of $A$. If rank($A$)=11, then $V$ is the only null vector of $A$, multiplied by a scaler. If on the other hand, rank($A$) $\leq$ 11, the solution to $X$ is the linear combinations of all null vectors of $A$. The null space of $A$ can be obtained by performing SVD on $A$, yielding

$$A^{m \times n} = U^{m \times m} D^{m \times n} (S^T)^{n \times n}$$
where $U$ and $S$ are orthonormal matrix and $D$ is a diagonal matrix, whose diagonal entries are called singular values and are arranged in descending order.

If $\text{rank}(A)=11$, the only null vector $A$ is the last column (corresponding to the smallest singular value) of $S$ matrix.

Alternatively, we can also solve $V$ by directly minimizing $||AV||^2_2$, which yields $(A^TA)V = 0$ or $(A^TA)V = \lambda V$, where $\lambda = 0$. As a result, solution to $V$ is the eigenvector of matrix $(A^TA)$, corresponding to zero eigenvalue. This implies that the eigenvectors of $A^TA$ correspond to the columns of the $S$ matrix.

Note $V$ is solved only up to a scale factor. The scale factor can be recovered using the fact that $||p_3||^2_2 = 1$. Let the null vector of $A$ be $V'$. The scale factor $\alpha = \sqrt{\frac{1}{V'^2(9)+V'^2(10)+V'^2(11)}}$. Hence, $V = \alpha V'$. 

17
Let $A = [B \ b]$, where

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

$$b = (-c_1 - r_1 \ldots - c_N - r_N)^T$$
\[
V = p_{34} \left( \begin{array}{c}
Y \\
1
\end{array} \right)
\]

\[
Y = \left( p_1^T p_{14}^T p_{24}^T p_3^T \right)^T / p_{34}
\]

Then, \( AV = p_{34} (BY + b) \). Since \( p_{34} \) is a constant, minimizing \( \|AV\|_2^2 \) is the same as minimizing \( \|BY + b\|_2^2 \), whose solution is \( Y = -(B^T B)^{-1} B^T b \). The rank of matrix B must be eleven.
The solution to $Y$ is up to a scale factor $p_{34}$. To recover the scale factor, we can use the fact that $|p_3| = 1$. 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

$$V = \begin{pmatrix} p_{34}Y \\ p_{34} \end{pmatrix}$$
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., \( \|p_3\|^2 = 1 \), then the problem is converted to a constrained linear least-squares problem. That is, minimize \( \|AV\|^2 \) subject to \( \|p_3\|^2 = 1 \).

\[
\epsilon^2 = \|AV\|^2 + \lambda(\|p_3\|^2 - 1)
\] (2)
Decomposing $A$ into two matrices $B$ and $C$, and $V$ into $Y$ and $Z$

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

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

$$B^{2N \times 9} = \begin{pmatrix}
M^T_1 & 1 & \bar{\delta} & 0 & -c_1 \\
\bar{\delta} & 0 & M^T_1 & 1 & -r_1 \\
\vdots & & & & \\
M^T_N & 1 & \bar{\delta} & 0 & -c_N \\
\bar{\delta} & 0 & M^T_N & 1 & -r_N
\end{pmatrix}$$

$$C^{2N \times 3} = \begin{pmatrix}
-c_1 M^T_1 \\
r_1 M^T_1 \\
\vdots \\
-c_N M^T_N \\
r_N M^T_N
\end{pmatrix}$$
\[ Y = \begin{pmatrix} p_1 \\ p_{14} \\ p_2 \\ p_{24} \\ p_{34} \end{pmatrix} \quad Z = p_3 \]
Then equation 2 can be rewritten as

$$
\epsilon^2 = \|BY + CZ\|_2^2 + \lambda(\|Z\|_2^2 - 1)
$$
Taking partial derivatives of $\epsilon^2$ with respect to $Y$ and $Z$ and setting them to zeros yield

$$\frac{\partial \epsilon^2}{\partial Y} = 0 \implies Y = -(B^T B)^{-1} B^T C Z$$

$$\frac{\partial \epsilon^2}{\partial Z} = 0 \implies C^T (I - B(B^T B)^{-1} B^T) C Z = \lambda Z$$

Apparently, the solution to $Z$ is the eigenvector of matrix $C^T (I - B(B^T B)^{-1} B^T) C$. Given $Z$, we can then obtain solution to $Y$. 
Substituting $Y$ into $||BY + CZ||^2$ leads to

\[
||BY + CZ||^2 \\
= ||B(B^TB)^{-1}B^TCZ + CZ||^2 \\
= ||(I - B(B^TB)^{-1}B^T)CZ||^2 \\
= Z^TC^T(I - B(B^TB)^{-1}B^T)^T(I - B(B^TB)^{-1}B^T)CZ \\
= Z^TC^T(I - B(B^TB)^{-1}B^T)CZ \\
= Z^T\lambda Z \\
= \lambda
\]

This proves that solution to $Z$ corresponds to the eigen vector of the smallest positive eigenvalue of matrix $C^T(I - B(B^TB)^{-1}B^T)C$.

Note

\[
(I - B(B^TB)^{-1}B^T)(I - B(B^TB)^{-1}B^T) = (I - B(B^TB)^{-1}B^T)
\]
Other Linear Techniques

• Another new method at


Note all linear methods we introduce belong to the category of direct linear transformation (DLT) formulation or the DLT algorithm.
Compute P: Unconstrained 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 the sum of the squared projection errors

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

First order methods, including gradient descent, Gauss-Newton method, and Levenberg Marquardt method, or second order method such as Newton method can be used to iteratively solve the project matrix.
Compute P: Unconstrained Non-linear Method

Let \( \hat{c}_i(p_1, p_3) = \frac{M_i^T p_1 + p_{14}}{M_i^T p_3 + p_{34}} \) and \( \hat{r}_i(p_2, p_3) = \frac{M_i^T p_2 + p_{24}}{M_i^T p_3 + p_{34}} \)

Let \( \hat{V}(p_1, p_2, p_3) = \begin{pmatrix} \hat{c}_1 \\ \hat{r}_1 \\ \vdots \\ \hat{c}_N \\ \hat{r}_N \end{pmatrix} \) and \( V = \begin{pmatrix} c_1 \\ r_1 \\ \vdots \\ c_N \\ r_N \end{pmatrix} \)

The objective function in Eq. 3 can be re-written as

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

\[
= (\hat{V} - V)^T(\hat{V} - V) \quad (4)
\]
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 \( ||p_3||_2^2 = 1 \). Introducing the lagrangian multiplier \( \lambda_1 \), we have the constrained objective function

\[
\epsilon^2 = \sum_{i=1}^{N}\left(\frac{M_i^T p_1 + p_{14}}{M_i^T p_3 + p_{34}} - c_i\right)^2 + \left(\frac{M_i^T p_2 + p_{24}}{M_i^T p_3 + p_{34}} - r_i\right)^2
+ \lambda_1 (||p_3||_2^2 - 1)
= \left(\hat{V} - V\right)^T(\hat{V} - V) + \lambda_1 (||p_3||_2^2 - 1)
\]  

(5)
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 $\|p_3\|^2_2 = 1$ and $(p_1 \times p_3) \cdot (p_2 \times p_3) = 0$, i.e., the full orthonormal constraints on $R$. 

Compute P: Constrained Non-linear Method 2
Introducing the lagrangian multipliers ($\lambda$s), we have the constrained objective function

$$
\epsilon^2 = \sum_{i=1}^{N} \left( \frac{M_i^T p_1 + p_{14}}{M_i^T p_3 + p_{34}} - c_i \right)^2 + \left( \frac{M_i^T p_2 + p_{24}}{M_i^T p_3 + p_{34}} - r_i \right)^2
+ \lambda_1 (\|p_3\|_2^2 - 1) + \lambda_2 (p_1 \times p_3) \cdot (p_2 \times p_3)
= (\hat{V} - V)^T (\hat{V} - V)
+ \lambda_1 (\|p_3\|_2^2 - 1) + \lambda_2 [(p_1 \times p_3) \cdot (p_2 \times p_3)]
$$

(8)
Non-linear Least-Squares (NLS) Solutions

The solutions to non-linear least-squares problems are typically iterative, starting from an initial guess.

- The gradient descent method (first order, applicable to any objective function)
- Gauss-Newton method (or Levenberg-Marquardt) (first order, applicable to only NLQ)
- The Newton method (second order, applicable to any objective function)

Given an initial $p_i$, non-linear methods iteratively update $p_i$, i.e.,

$$p_i^t = p_i^{t-1} + \eta \Delta p_i$$

They differ in the way $\Delta p_i$ is computed.
Gradient Descent Method

First order gradient descent may be used to solve $P$ iteratively. For objective function $8$, we can solve $p_3$ as follows

$$p^t_3 = p^{t-1}_3 + \eta \Delta p_3$$

where $\eta$ is the learning rate and $\Delta p_3 = -\nabla p_3 \epsilon^2$. It can be computed as follows

$$\nabla p_3 \epsilon^2 = 2 \frac{\partial \hat{V}}{\partial p_3} (\hat{V} - V)$$

$$+ 2 \lambda_1 p_3 + \lambda_2 \left[ \frac{\partial (p_1 \times p_3)}{\partial p_3} \cdot (p_2 \times p_3) \right]$$

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

\[
\frac{\partial \hat{V}}{\partial p_3} = \left( \frac{\partial \hat{c}_1}{\partial p_3} \quad \frac{\partial \hat{r}_1}{\partial p_3} \quad \ldots \quad \frac{\partial \hat{c}_N}{\partial p_3} \quad \frac{\partial \hat{r}_N}{\partial p_3} \right)
\]

And

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

\(\frac{\partial p_3}{\partial p_3}\) is a 3 × 3 identity matrix. \(p_1 \times \frac{\partial p_3}{\partial p_3}\) is cross product between a vector and a matrix, which can be carried out by cross product of the vector with each column of the matrix, yielding another 3 × 3 matrix. Similarly, we compute \(\frac{\partial (p_2 \times p_3)}{\partial p_3}\). We can follow the same to compute \(p_1\), \(p_{14}\), \(p_2\), \(p_{24}\) and \(p_{34}\).
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.
Gauss-Newton Method

For unconstrained non-linear least squares problem, the object function is \( \epsilon^2 = ||\hat{V} - V||^2_2 \). We can apply the Gauss-Newton method. For \( p_3 \) in Eq. 4, we have

\[
p_3^t = p_3^{t-1} + \eta \Delta p_3
\]

where \( \Delta p_3 \) in Eq. 4 can be computed as follows

\[
\Delta p_3 = -\left[ \left( \frac{\partial \hat{V}}{\partial p_3} \right) \left( \frac{\partial \hat{V}}{\partial p_3} \right)^T \right]^{-1} \left( \frac{\partial \hat{V}}{\partial p_3} \right) (\hat{V} - V)
\]

We can similarly solve for \( p_1 \) and \( p_2 \).
For constrained non-linear least squares problem, let $C(p_3)$ be the equality constraint on $p_3$. The objective function changes to

$$
\epsilon^2 = ||\hat{V} - V||_2^2 + \lambda C(p_3)
$$

The Gauss-Newton method can be changed as follows. For $p_3$ in above equation, we have

$$
p_3^t = p_3^{t-1} + \eta \Delta p_3
$$

where $\Delta p_3$ in Eq. 4 can be computed as follows

$$
\Delta p_3 = [(\frac{\partial \hat{V}}{\partial p_3})(\frac{\partial \hat{V}}{\partial p_3})^T]^{-1}(\frac{\partial \hat{V}}{\partial p_3})[-(\hat{V} - V) + \lambda \frac{\partial C}{\partial p_3}]
$$

We can similarly solve for $p_1$ and $p_2$. 

Gauss-Newton Method

Detailed derivations of Gauss-Newton method:

First order Taylor expansion of $\hat{V}$ w.r.t. to current $p^0_3$, yields

$$\hat{V}(p_3) = \hat{V}(p^0_3) + \left( \frac{\partial \hat{V}(p^0_3)}{\partial p_3} \right)^T \Delta p_3$$

Substituting it into Eq. 4 yields

$$\epsilon^2 = [\hat{V}(p^0_3) + \left( \frac{\partial \hat{V}(p^0_3)}{\partial p_3} \right)^T \Delta p_3 - V]^T [\hat{V}(p^0_3) + \left( \frac{\partial \hat{V}(p^0_3)}{\partial p_3} \right)^T \Delta p_3 - V]$$

Taking partial derivative of $\epsilon^2$ wrt to $\Delta p_3$ and the setting the result to zero yields

$$\frac{\partial \epsilon^2}{\partial \Delta p_3} = 2 \frac{\partial \hat{V}(p^0_3)}{\partial p_3} \left( \frac{\partial \hat{V}(p^0_3)}{\partial p_3} \right)^T \Delta p_3 - V = 0$$
Solving $\Delta p_3$ yields

$$\Delta p_3 = -\left[(\frac{\partial \hat{V}(p_3^0)}{\partial p_3})(\frac{\partial \hat{V}(p_3^0)}{\partial p_3})^T\right]^{-1}(\frac{\partial \hat{V}(p_3^0)}{\partial p_3})(\hat{V}(p_3^0) - V)$$

With constraint $C(p_3)$,

$$\epsilon^2 = [\hat{V}(p_3^0) + (\frac{\partial \hat{V}(p_3^0)}{\partial p_3})^T \Delta p_3 - V]^T[\hat{V}(p_3^0) + (\frac{\partial \hat{V}(p_3^0)}{\partial p_3})^T \Delta p_3 - V] + \lambda[C(p_3^0) + (\frac{\partial C(p_3^0)}{\partial p_3})^T \Delta p_3]$$

Hence,

$$\Delta p_3 = \left[(\frac{\partial \hat{V}}{\partial p_3})(\frac{\partial \hat{V}}{\partial p_3})^T\right]^{-1}\left(\frac{\partial \hat{V}}{\partial p_3}\right)[- (\hat{V} - V) + \lambda \frac{\partial C}{\partial p_3}]$$
Levenberg Marquardt (LM) algorithm

LM algorithm is an improved version of Gauss-Newton method, i.e., for $p_3$ in Eq. 4, we have

$$p_3^t = p_3^{t-1} + \eta \Delta p_3$$

where $\Delta p_3$ is computed differently

Unconstrained LM:

$$\Delta p_3 = -[\left(\frac{\partial \hat{V}}{\partial p_3}\right)\left(\frac{\partial \hat{V}}{\partial p_3}\right)^T + \alpha I]^{-1}\left(\frac{\partial \hat{V}}{\partial p_3}\right)(\hat{V} - V)$$

Constrained LM:

$$\Delta p_3 = \left[\left(\frac{\partial \hat{V}}{\partial p_3}\right)\left(\frac{\partial \hat{V}}{\partial p_3}\right)^T + \alpha I\right]^{-1}\left(\frac{\partial \hat{V}}{\partial p_3}\right)[-(\bar{V} - V) + \lambda \frac{\partial C}{\partial p_3}]$$

where $I$ is an identity matrix and $\alpha$ is a damping factor that varies with each iteration. The iteration starts with a large $\alpha$ and
gradually reduces $\alpha$ value as the iteration goes. With a small $\alpha$, the LM algorithm becomes Gauss-Newton method and becomes the gradient descent method with a large $\alpha$. 
Unconstrained Newton Method

Newton method is the second order method for mainly solving unconstrained optimization. It assumes the objective function is second order differentiable. Given the objective function

$$\epsilon^2 = (\hat{V} - V)^T(\hat{V} - V)$$

Take second order Taylor expansion of $\epsilon^2$ wrt to $p_0^3$ yields

$$\epsilon^2 = (\hat{V}(p_0^3) - V)^T(\hat{V}(p_0^3) - V) + \left(\frac{\partial \epsilon^2(p_0^3)}{\partial p_3}\right)^T \Delta p_3$$

$$+ \frac{1}{2} \left(\frac{\partial^2 \epsilon^2(p_0^3)}{\partial p_3^2}\right) \Delta p_3 (\Delta p_3)^T$$

$$\frac{\partial \epsilon^2}{\partial \Delta p_3} = \frac{\partial \epsilon^2(p_0^3)}{\partial p_3} + \left(\frac{\partial^2 \epsilon^2(p_0^3)}{\partial p_3^2}\right) \Delta p_3 = 0$$
Hence

\[
\Delta p_3 = -(\frac{\partial^2 \epsilon^2(p_3^0)}{\partial p_3^2})^{-1}(\frac{\partial \epsilon^2(p_3^0)}{\partial p_3})
\]

where \( \frac{\partial^2 \epsilon^2}{\partial p_3^2} \) is the Hessian matrix and \( \frac{\partial \epsilon^2}{\partial p_3} \) is the Jacobian matrix.

\[
p_3^t = p_3^{t-1} + \eta \Delta p_3
\]
Constrained Newton Method

Given the constrained objective function with constraint $C(p_3)$ on $p_3$

$$\epsilon^2 = (\hat{V} - V)^T(\hat{V} - V) + \lambda C(p_3)$$

Take second order Taylor expansion of $\epsilon^2$ wrt to $p_0^3$ yields

$$\epsilon^2 = (\hat{V}(p_0^3) - V)^T(\hat{V}(p_0^3) - V) + \left(\frac{\partial \epsilon^2(p_0^3)}{\partial p_3}\right)^T \Delta p_3$$

$$+ \frac{1}{2} \left(\frac{\partial^2 \epsilon^2(p_0^3)}{\partial p_3^2}\right)^T (\Delta p_3)(\Delta p_3)^T$$

$$+ \lambda \left[C(p_0^3) + \left(\frac{\partial C(p_0^3)}{\partial p_3}\right)^T \Delta p_3 + \frac{1}{2} \left(\frac{\partial^2 C(p_0^3)}{\partial p_3^2}\right)^T (\Delta p_3)(\Delta p_3)^T\right]$$
\[
\frac{\partial \epsilon^2}{\partial \Delta p_3} = \frac{\partial \epsilon^2(p^0_3)}{\partial p_3} + \left( \frac{\partial^2 \epsilon^2(p^0_3)}{\partial p^2_3} \right) \Delta p_3 \\
+ \lambda \frac{\partial C(p^0_3)}{\partial p_3} + \lambda \left( \frac{\partial^2 C(p^0_3)}{\partial p^2_3} \right) \Delta p_3 = 0
\]

Hence

\[
\Delta p_3 = -\left[ \frac{\partial^2 \epsilon^2(p^0_3)}{\partial p^2_3} + \lambda \frac{\partial^2 C(p^0_3)}{\partial p^2_3} \right]^{-1} \left[ \frac{\partial \epsilon^2(p^0_3)}{\partial p_3} + \lambda \frac{\partial C(p^0_3)}{\partial p_3} \right]
\]

\[
p^{t}_3 = p^{t-1}_3 + \lambda \Delta p_3
\]
Another way of solving this problem is to perform minimization directly with respect to the intrinsic and extrinsic parameters.
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 $N$ ($K > 6$) points 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.

- Step 4: Using the subset of points selected in Step 3 plus all of the other inlier points which contributed to the largest count, recompute the best \( P \) for all of these points.

More information on the RANSAC method may be found at [https://opencv.org/evaluating-opencvs-new-ransacs/](https://opencv.org/evaluating-opencvs-new-ransacs/)

\(^a\)the exact number of times is determined by the required probability that one of subset does not contain the outliers
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.
Compute Camera Parameters from $P$

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

\[
    r_3 = p_3 \\
    c_0 = p_1 p_3^T \\
    s_x f = \sqrt{p_1 p_1^T - c_0^2} = ||p_1^T \times p_3^T||_2 \\
    s_y f = \sqrt{p_2 p_2^T - r_0^2} = ||p_2^T \times p_3^T||_2 \\
    t_x = (p_{14} - c_0 t_z)/(s_x f) \\
    t_y = (p_{24} - r_0 t_z)/(s_y f)
\]
\[ \mathbf{r}_1 = \frac{(p_1 - c_0 r_3)}{(s_x f)} \]

\[ \mathbf{r}_2 = \frac{(p_2 - r_0 r_3)}{(s_y f)} \]
Compute Camera Parameters from $P$ (cont’d)

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

Following the RQ decomposition algorithm (different from the well-known QR decomposition) in https://math.stackexchange.com/posts/1640762/revisions

We can decompose $P_3$ into $P_3 = AB$, where $A$ is an upper triangle matrix and $B$ is an orthonormal matrix.

Hence, we have $W = A$ and $R = B$.

Given $W$, $T$ can be recovered as $T = W^{-1}P_4$, where $P_4$ is the last column of $P$. 
Approximate solution to imposing orthonormality

Let \( \hat{R} \) be the estimated rotation matrix \( R \). It is not orthonormal. We can find an orthonormal matrix \( R \) 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 \( \hat{R} = UIV^T \). \( \hat{R} \) is the orthonormal matrix that is closest to \( \hat{R} \).
Estimating Scale Factor using Vanishing Points

Let \((c_1, r_1)\) and \((c_2, r_2)\) be the vanishing points for two sets of parallel lines that are orthogonal to each other. Assuming the image center \((c_0, r_0)\) is known (i.e., at the image center), and the horizontal and vertical scale factors for horizontal are the same, i.e., \(fs_x = fs_y = fs\). We can easily prove that

\[(fs)^2 = (c_1 - c_0)(c_2 - c_0) + (r_1 - r_0)(r_2 - r_0)\]

The derivations of the equation above require: 1) \(c_i = \frac{fsN_i^x}{N_z^i}\) and \(r_i = \frac{fsN_i^y}{N_z^i}\), where \(i=1,2\) and \(N_i = (N_x^i, N_y^i, N_z^i)^T\) is the orientation of the \(i\)th line; and 2) \((N^1)^T(N^2) = 0\).
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 $i_i$. 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

\[ \Theta = (c_0 \ r_0 \ f \ s_x \ s_y \ \phi \ \kappa \ t_x \ t_y \ t_z)^T \]

\[ g(\Theta, M_i) = \frac{M_i^T p_1 + p_{14}}{M_i^T p_3 + p_{34}} \]

\[ f(\Theta, M_i) = \frac{M_i^2 p_2 + p_{24}}{M_i^T p_3 + p_{34}} \]

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.,

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

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.

Organize the objective function in matrix format and then introduce the gradient descent, the Gauss-Newton, and the Levenberg-Marquardt method directly with respect to the camera parameters.
Calibration with Planar Object

Since planar points reduces the rank of A matrix to 8, we cannot uniquely solve for \( V \) using planar object as the solution to \( V \) is a linear combination of the 4 null vectors of \( A \) matrix (up to four scale factors).

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 \( W \).
For planar calibration object, we can place the object frame on the plane such that $z=0$. Hence, given the $i$th 2D/3D point, we have

$$
\lambda_i \begin{pmatrix}
c_i \\
r_i \\
1
\end{pmatrix} = W \begin{pmatrix}
r_1 & r_2 & T
\end{pmatrix} \begin{pmatrix}
x_i \\
y_i \\
1
\end{pmatrix}
$$

$$
= H \begin{pmatrix}
x_i \\
y_i \\
1
\end{pmatrix}
$$

(9)

where $r_1, r_2$, and $T$ are respectively the first and second columns
of the rotation matrix, and the translation vector. Given a minimum of 4 points, we can solve $H$ using either linear or non-linear methods. Given $H$, we have

$$Wr_1 = h_1$$
$$Wr_2 = h_2$$

(10)

Eq. 10 can be equivalently written as

$$r_1 = W^{-1}h_1$$
$$r_2 = W^{-1}h_2$$

(11)

Using the fact that $r_1$ and $r_2$ are unit vectors and they are orthogonal to each other, we can derive the following

$$||r_1||^2 = 1 \Rightarrow h_1^T W^{-t} W^{-1} h_1 = 1$$
$$||r_2||^2 = 1 \Rightarrow h_2^T W^{-t} W^{-1} h_2 = 1$$
\[ r_1^T r_2 = 0 \Rightarrow h_1^T W^{-t} W^{-1} h_2 = 0 \] (12)

Let \( K = W^{-t} W^{-1} \), Eq. 12 can be re-rewritten as

\[
\begin{align*}
    h_1^T K h_1 &= 1 \\
    h_2^T K h_2 &= 1 \\
    h_1^T K h_2 &= 0
\end{align*}
\] (13)

Eq. 13 provides three equations for \( K \). We need another image to produce another three equations. In total, we have six equations to solve for six unknowns in \( K \) as \( K \) is symmetric. Given \( K \), we can then apply matrix decomposition to solve for \( W \). For details see Zhengyou Zhang’s calibration method at http://research.microsoft.com/en-us/um/people/zhang/Calib/
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.
Line Basics

• In general, a line equation can be expressed as \( p = p_0 + \lambda N \), where \( p \) is an any point on the line, \( p_0 \) is a given point on the line, \( N \) is the orientation of the line, and \( \lambda \) is scalar that measures the distance between \( p \) and \( p_0 \). The parameters of the line include \( p_0 \) and \( N \). If \( p_0 \) is chosen such that \( Op_0 \) is perpendicular to the line, we can reduce the number of line parameters by one, where \( O \) is the origin. This line equation applies to both 2D and 3D line.

• Alternatively, given two points \( p_1 \) and \( p_2 \), the equation for the line that goes through the two points is \( p = \alpha p_1 + (1 - \alpha) p_2 \).

• A 2D line equation in row-column frame can also be expressed as \( \alpha c + \beta r + \gamma = 0 \), where \((\alpha, \beta, \gamma)\) are the line parameters. The orientation of the line is \( \left( \frac{\alpha}{\sqrt{\alpha^2 + \beta^2}}, \frac{-\beta}{\sqrt{\alpha^2 + \beta^2}} \right) \).
• An image line $\alpha c + \beta r + \gamma = 0$ in row-column frame, together with the camera center, forms a plane (backprojection plane) and the equation of the plane in camera frame is

$$[W^T \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}]^T \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} = 0.$$ 

The intersection of this plane with the image plane $z_c = f$ gives the equation of the image plane in camera frame.

• Given two points $p_1$ and $p_2$ in image plane in homogeneous coordinates, the line between them can be computed as $l = p_1 \times p_2$, where $\times$ stands for the cross product and $l$ is a $3 \times 1$ vector that encodes image line parameters $(\alpha, \beta, \gamma)$.

• The intersection of two lines in 2D can be computed from the cross product of their line parameters. Let $l = (\alpha, \beta, \gamma)^T$ and
$l' = (\alpha', \beta', \gamma')^T$ be the parameters of two 2D lines, their intersection point $p$ can be computed as $p = l \times l'$. The intersection point coordinates can be computed by dividing the first two elements of $p$ by the third element.

- Given a 3D line $L = P_0 + \lambda N$ and the corresponding image line $l = (\alpha, \beta, \gamma)^T$ in row-column coordinate frame, the equation of the projection plane that contains both the 3D line and 2D line in camera frame is $n^T \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} = 0$, where $n$ is the normal of the plane in camera frame and

$$n = W^T \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}.$$
Camera Calibration using Lines

Let $M = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ be a 3D point w.r.t the object frame and $m = \begin{pmatrix} c \\ r \end{pmatrix}$ be the corresponding image point in row-column frame.

Let the 3D line be parameterized as

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} + \lambda \begin{pmatrix} N_x \\ N_y \\ N_z \end{pmatrix} = M_3 + \alpha N \quad (1)$$

and the corresponding image line in the row-column frame be parameterized as

$$ac + br + \gamma = 0 \Rightarrow \begin{pmatrix} c \\ r \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \\ 1 \end{pmatrix} n \quad (2)$$

Let $m_0 = \begin{pmatrix} c_0 \\ r_0 \end{pmatrix}$ be corresponding image point for $M_3$. From the pair of $m_0$ and $M_3$, we have

$$\begin{pmatrix} c_0 \\ r_0 \\ 1 \end{pmatrix} = P \begin{pmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{pmatrix} \quad (3)$$

where $P$ is the projection matrix. Combining Eqs (1), (2), and (3) produces

$$N'R'W'n = 0 \quad (4)$$

See appendix for details.

Eqs (3) and (4) provide a total of three equations for $P$. A minimum of 4 pairs of 2D/3D lines are needed to solve $P$ linearly. We can impose one orthonormal constraint $\|p3\|^2 = 1$ to still have a linear solution.

Question: why do we need 4 pairs of 2D/3D lines instead of 3
pairs as each line is determined by two points and 3 pairs will provide 6 points, which is enough to recover $P$ using point-based methods. Hint: consider the degree of freedoms for a 3D line and that of two 3D points. A 3D line has 5 DOF, while 2 3D points have 6 DOF.
Camera Calibration using Lines

Eq. 4 is derived as follows.

Incorporating the full projection equation, Eq. 1 and Eq. 3 into Eq. 2 yields

\[
\begin{pmatrix}
c' \\
r' \\
1
\end{pmatrix}
\mathbf{n} = 0 \Rightarrow [\mathbf{P}]
\begin{pmatrix}
x' \\
y' \\
z'
\end{pmatrix}
= \begin{pmatrix}
x_0 \\
y_0 \\
z_0
\end{pmatrix}
+ \lambda \mathbf{P}
\begin{pmatrix}
x' \\
y' \\
z'
\end{pmatrix}
= \begin{pmatrix}
x_0 \\
y_0 \\
z_0
\end{pmatrix}
+ \lambda \mathbf{P}
\begin{pmatrix}
x' \\
y' \\
z'
\end{pmatrix}
\mathbf{n} = 0
\Rightarrow [\mathbf{W}]\mathbf{n} = 0
\Rightarrow [\mathbf{WR}]\mathbf{n} = 0
\]

Note given \( \lambda \begin{pmatrix}
c \\
r \\
1
\end{pmatrix} = \begin{pmatrix}
x_c \\
y_c \\
z_c
\end{pmatrix} \) and substituing it into the
The line equation yields
\[
\begin{bmatrix}
    x_c \\
    y_c \\
    z_c
\end{bmatrix}^T W^T \begin{bmatrix}
    \alpha \\
    \beta \\
    \gamma
\end{bmatrix} = 0
\]

This is the equation for the backprojection plane w.r.t to the camera frame that contains the 2D/3D lines. Its normal is
\[
W^T \begin{bmatrix}
    \alpha \\
    \beta \\
    \gamma
\end{bmatrix}
\]

Therefore
\[
||W^T \begin{bmatrix}
    \alpha \\
    \beta \\
    \gamma
\end{bmatrix}||_2
\]
Camera Calibration 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 \(M\). 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, $M = \begin{pmatrix} \frac{fs_x r_1}{\bar{z}_c} \\ \frac{fs_y r_2}{\bar{z}_c} \end{pmatrix}$, we have

$$\frac{fs_x}{\bar{z}_c} = ||m_1||_2$$

$$\frac{fs_y}{\bar{z}_c} = ||m_2||_2$$

where $m_1$ and $m_2$ are the first row and the second row of the $M$ matrix.

Then,

$$r_1 = \frac{m_1}{||m_1||_2}$$
\[ \mathbf{r}_2 = \frac{m_2}{\|m_2\|_2} \]

\[ \mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2 \]

With weak perspective projection, we need a minimum of 3 points. But we can only solve the above parameters, i.e., the \( \frac{f s_x}{z_c} \), \( \frac{f s_y}{z_c} \), and the orientation of the object frame relative to the camera frame.
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 \textsuperscript{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

\textsuperscript{a}the paper appears in CVPR 01, pages 125-132
camera true distortion model. Hence,

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

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

\[
D_1 = \begin{pmatrix}
x & y & z & 1 & 0 & 0 & 0 & 0 & -\hat{c}x & -\hat{c}y & -\hat{c}z & -c_0 \\
0 & 0 & 0 & 0 & x & y & z & 1 & -\hat{r}x & -\hat{r}y & -\hat{r}z & -r_0
\end{pmatrix}
\]
The above equation can be solved as a polynomial eigen value problem. MATLAB function `polyeig` can be used to obtain the solution for both $k$ and $V$. To use `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 `polyeig`

$$(D_1^T D_1 + k(D_1^T D_2 + D_2^T D_1) + k^2 D_2^T D_2)V = 0^b$$

The solution, however, assumes the knowledge of the image center. We can fix it as the center of the image. Study shows that $^b$when $k$ is small, $D_1$ is close to the A matrix.
the precise location of the distortion center does not strongly affect the correction (see Ref. 9 of Fitz’s paper).

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.
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 viewpoint but different view directions)
- Only translational movement (different view points but same view direction)
Camera Self-Calibration With Only Rotation

As we do not have 3D information, we can locate the object frame anywhere 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 indexed by $i$. Let $j$ index to the $j$th image point.
$X_0-Y_0-Z_0$ — original camera frame before rotation

$X-Y-Z$ — be the object frame.

Assume $X-Y-Z$ and $X_0-Y_0-Z_0$ coincide.
\[
\lambda_{0,j} \begin{pmatrix} c_{0,j} \\ r_{0,j} \\ 1 \end{pmatrix} = WM_0 \begin{pmatrix} X_j \\ Y_j \\ Z_j \\ 1 \end{pmatrix} = W[I \ 0] \begin{pmatrix} X_j \\ Y_j \\ Z_j \\ 1 \end{pmatrix} = W \begin{pmatrix} X_j \\ Y_j \\ Z_j \end{pmatrix}
\]

(14)

If we rotate the camera frame from the reference by a rotation matrix \( R_i \), we have

\[
\lambda_{i,j} \begin{pmatrix} c_{i,j} \\ r_{i,j} \\ 1 \end{pmatrix} = WM_i \begin{pmatrix} X_j \\ Y_j \\ Z_j \\ 1 \end{pmatrix} = W[R_i \ 0] \begin{pmatrix} X_j \\ Y_j \\ Z_j \\ 1 \end{pmatrix} = WR_i \begin{pmatrix} X_j \\ Y_j \\ Z_j \end{pmatrix}
\]

(15)
Denote $\lambda'_{i,j} = \frac{\lambda_{0,j}}{\lambda_{i,j}}$, substituting 14 to 15 to remove $(X_j, Y_j, Z_j)^T$ yields

$$
\begin{pmatrix}
c_{i,j} \\
r_{i,j} \\
1
\end{pmatrix}
= \lambda'_{i,j} \mathbf{W} \mathbf{R}_i \mathbf{W}^{-1}
\begin{pmatrix}
c_{0,j} \\
r_{0,j} \\
1
\end{pmatrix}
$$

(16)

Let $\mathbf{B}_i = \mathbf{W} \mathbf{R}_i \mathbf{W}^{-1} = \begin{pmatrix}
B_{i11} & B_{i12} & B_{i13} \\
B_{i21} & B_{i22} & B_{i23} \\
B_{i31} & B_{i32} & B_{i33}
\end{pmatrix}$, we have

$$
\begin{pmatrix}
c_{i,j} \\
r_{i,j} \\
1
\end{pmatrix}
= \lambda_{i,j}
\begin{pmatrix}
B_{i11} & B_{i12} & B_{i13} \\
B_{i21} & B_{i22} & B_{i23} \\
B_{i31} & B_{i32} & B_{i33}
\end{pmatrix}
\begin{pmatrix}
c_{0,j} \\
r_{0,j} \\
1
\end{pmatrix}
$$

(17)
This leads to three equations

\[ c_{i,j} = \lambda_{i,j} (B_{i11} c_{0,j} + B_{i12} r_{0,j} + B_{i13}) \]
\[ r_{i,j} = \lambda_{i,j} (B_{i21} c_{0,j} + B_{i22} r_{0,j} + B_{i23}) \]
\[ 1 = \lambda_{i,j} (B_{i31} c_{0,j} + B_{i32} r_{0,j} + B_{i33}) \]  

(18)

Since \( \lambda_{i,j} = 1/(B_{i11} c_{0} + B_{i12} r_{0} + B_{i13}) \), substituting \( \lambda_{i,j} \) to the above equations yields

\[ c_{i} (B_{i31} c_{0} + B_{i32} r_{0} + B_{i33}) = (B_{i11} c_{0} + B_{i12} r_{0} + B_{i13}) \]
\[ r_{i} (B_{i31} c_{0} + B_{i32} r_{0} + B_{i33}) = (B_{i21} c_{0} + B_{i22} r_{0} + B_{i23}) \]

Given \( N \) (j=1,2, ...,N) points, we can setup 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 2
equations by $B_{i33}$ and they can be rewritten in matrix format

$$
\begin{pmatrix}
  -c_0 & -r_0 & 1 & 0 & 0 & 0 & c_i c_0 & c_i r_0 \\
  0 & 0 & 0 & -c_0 & -r_0 & 1 & r_i c_0 & r_i r_0 \\
\end{pmatrix}
\begin{pmatrix}
  b_i \\
\end{pmatrix} =
\begin{pmatrix}
  -c_i \\
  -r_i \\
\end{pmatrix}
$$

(19)

where $b_i = (B_{i_{11}} B_{i_{12}} B_{i_{13}} B_{i_{21}} B_{i_{22}} B_{i_{23}} B_{i_{31}} B_{i_{32}})^T / B_{i33}$. If we know at least 4 points in two images (such as $i = 0, 1$), we can solve for $b_i$ up to a scale factor. The scale factor can be solved using the fact that the determinant of $WR_iW^{-1}$ is unit.

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

To solve for $W$ with a unknown $R_i$. From $B_iW = WR_iW^{-1}$, we have

$$
R_i = W^{-1}B_iW \quad R_i^{-T} = W^TB_i^{-1}W^{-T}
$$
Since $R = R^{-T}$, therefore we have

\[(WW^T)B_i^{-T} = B_i(WW^T) \quad (20)\]

Assume $C = WW^T$, we have

\[
C = \begin{pmatrix}
  s_x f & 0 & c_0 \\
  0 & s_y f & r_0 \\
  0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
  s_x f & 0 & 0 \\
  0 & s_y f & 0 \\
  c_0 & r_0 & 1
\end{pmatrix}
\]

\[
= \begin{pmatrix}
  s_x^2 f^2 + c_0^2 & c_0 r_0 & c_0 \\
  c_0 r_0 & s_y^2 f^2 + r_0^2 & r_0 \\
  c_0 & r_0 & 1
\end{pmatrix} \quad (21)
\]

Equation 20 can be rewritten

\[B_i CB_i^T - C = 0 \quad (22)\]
Since $C$ is symmetric, Eq. 22 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 22 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 $W$ directly from Choleski factorization. Alternatively, we can also 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.

Section 11.1.3 of [1] introduced a method to recover the focus length from a single rotation, assuming the principal point is located in the image center.
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

\[
\lambda_{0,j} \begin{pmatrix} c_{0,j} \\ r_{0,j} \\ 1 \end{pmatrix} = W[I T_0] \begin{pmatrix} X_j \\ Y_j \\ Z_j \\ 1 \end{pmatrix} = W \begin{pmatrix} X_j \\ Y_j \\ Z_j \end{pmatrix} \tag{23}
\]
\[ \lambda_{i,j} \begin{pmatrix} c_{i,j} \\ r_{i,j} \\ 1 \end{pmatrix} = W \left[ I \ T_i \right] \begin{pmatrix} X_j \\ Y_j \\ Z_j \\ 1 \end{pmatrix} = W \begin{pmatrix} X_j \\ Y_j \\ Z_j \\ 1 \end{pmatrix} + WT_i \quad (24) \]

From the above equations, we have

\[ \lambda_{i,j} \begin{pmatrix} c_{i,j} \\ r_{i,j} \\ 1 \end{pmatrix} = \lambda_{0,j} \begin{pmatrix} c_{0,j} \\ r_{0,j} \\ 1 \end{pmatrix} + WT_i \quad (25) \]
Equation 25 can be rewritten

\[
\begin{pmatrix}
\lambda_{i,j} \\
\lambda_{0,j}
\end{pmatrix}
\begin{pmatrix}
c_{i,j} \\
r_{i,j} \\
1
\end{pmatrix}
= \begin{pmatrix}
\lambda_{0,j} \\
\lambda_{0,j}
\end{pmatrix}
\begin{pmatrix}
c_{0,j} \\
r_{0,j} \\
1
\end{pmatrix}
+ \begin{pmatrix}
s_x f & 0 & c_0 \\
0 & s_y f & r_0 \\
0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
t_{i,x} \\
t_{i,y} \\
t_{i,z}
\end{pmatrix}
\]

(26)

then, we get three equations, assuming \( T \) is known \(^{a}\)

\[
\begin{align*}
\lambda_{i,j}c_{i,j} &= \lambda_{0,j}c_{0,j} + (s_x ft_{i,x} + c_0 t_{i,z}) \\
\lambda_{i,j}r_{i,j} &= \lambda_{0,j}r_{0,j} + (s_y ft_{i,y} + r_0 t_{i,z}) \\
\lambda_{i,j} &= \lambda_{0,j} + t_{i,z}
\end{align*}
\]

(27)

Substituting \( \lambda_{i,j} = \lambda_{0,j} + t_{i,z} \) to the above two equations yields

\[
\begin{align*}
\lambda_{0,j}c_{i,j} + t_{i,z}c_{i,j} &= \lambda_{0,j}c_{0,j} + s_x ft_{i,x} + c_0 t_{i,z} \\
\end{align*}
\]

\(^{a}\)there is no linear solution if \( T \) is unknown
\[
\lambda_{0,j} r_{i,j} + t_{iz} r_{i,j} = \lambda_{0,j} r_{0,j} + s_y f t_{iy} + r_0 t_{iz}
\]  

(28)

Combining the two equations above to remove \( \lambda_{0,j} \) yields

\[
(r_{ij} - r_{0,j}) t_{ix} f s_x - (c_{ij} - c_{0,j}) t_{iy} f s_y +
(r_{i,j} - r_{0,j}) t_{iz} c_0 - (c_{i,j} - c_{0,j}) t_{iz} r_0
= (r_{ij} - r_{0,j}) t_{iz} c_{i,j} - (c_{i,j} - c_{0,j}) t_{iz} r_{i,j}
\]

where \( \mathbf{W}' = (s_x f \quad s_y f \quad c_0 \quad r_0)^T \). If we know at least 4 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 (\( \text{rank}=3 \)) since the first and third columns as well as the second and fourth columns are different only by a scale factor given the same translation. Therefore, one translation is not enough.
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, photogrammetry, 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, depending on if W is known or not.

Case 1: 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.

Case 2: 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 $P$. Alternatively, we can assume some prior knowledge about $W$ such as the horizontal and vertical scale factors are equal, i.e., $f s_x = f s_y$ and the principal point is located in the image center. This knowledge can be imposed as constraints during the estimation.
Alternatively, we can set it up as a non-linear optimization problem, with the constraint of $R^{-1} = R^T$ imposed, along with any prior knowledge on $W$ as discussed in the last slide.
For weak perspective projection, we have

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

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 \(M\). A minimum of 3 points are enough to uniquely solve for the matrix \(M\).
Given $M$ and the parameterization for $M$ introduced in the previous chapter, we have

\[
\frac{fs_x}{z_c} = |m_1|
\]

\[
\frac{fs_y}{z_c} = |m_2|
\]

where $m_1$ and $m_2$ are the first row and the second row of the $M$ matrix.

Then,

\[
r_1 = \frac{m_1}{|m_1|}
\]

\[
r_2 = \frac{m_2}{|m_2|}
\]

\[
r_3 = r_1 \times r_2
\]
Pose Estimation Under Weak Perspective Projection

Given $R$, assuming $W$ is known (i.e., calibrated camera), $T$ can be solved using

$$
\lambda W^{-1} \begin{pmatrix} c \\ r \\ 1 \end{pmatrix} = R \begin{pmatrix} x \\ y \\ z \end{pmatrix} + T
$$

If $W$ is unknown, under weak projection, we have

$$
\bar{z}_c \begin{pmatrix} c \\ r \\ 1 \end{pmatrix} = \begin{pmatrix} f s_x r_1 & f s_x t_x + c_0 \bar{z}_c \\ f s_y r_2 & f s_y t_y + r_0 \bar{z}_c \\ 0 & \bar{z}_c \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}
$$
Dividing both sides by $\bar{z}_c$ yields

\[
\begin{pmatrix}
c \\
r
\end{pmatrix}
= M_{2\times3}
\begin{pmatrix}
x \\
y \\
z
\end{pmatrix} +
\begin{pmatrix}
v_x \\
v_y
\end{pmatrix}
\]

where $M_{2\times3}$ is the first $2 \times 3$ submatrix, $v_x = \frac{p_{14}}{p_{34}} = \frac{fs_x}{\bar{z}_c} t_x + c_0$ and $v_y = \frac{p_{24}}{p_{34}} = \frac{fs_x}{\bar{z}_c} t_y + r_0$. 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. We can never solve for $t_z$.

With weak perspective projection, we need a minimum of 3 points. We can solve R but T up to a scale factor.
References