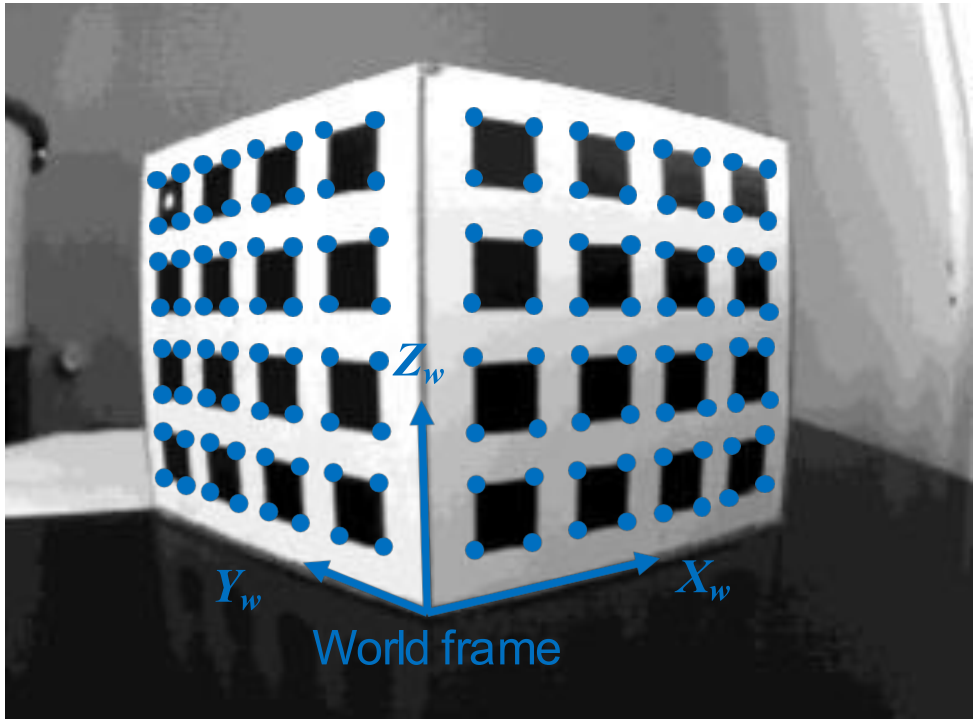
Tsai's Method: Calibration from 3D Objects

- This method was proposed in 1987 by Tsai and consists of measuring the 3D position of $n \geq 6$ **control points** on a 3D calibration target and the **2D coordinates of their projection** in the image.



Tsai, Roger Y. (1987) "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses," *IEEE Journal of Robotics and Automation*, 1987. PDF.

Applying the Direct Linear Transform (DLT) algorithm

The idea of the DLT is to rewrite the perspective projection equation as a **homogeneous linear equation** and solve it by standard methods. Let's write the perspective equation for a generic 3D-2D point correspondence:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K\left[R \mid T\right] \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad \Rightarrow$$

$$\Rightarrow \quad \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

$$\Rightarrow \quad \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u r_{11} + u_0 r_{31} & \alpha_u r_{12} + u_0 r_{32} & \alpha_u r_{13} + u_0 r_{33} & \alpha_u t_1 + u_0 t_3 \\ \alpha_v r_{21} + v_0 r_{31} & \alpha_v r_{22} + v_0 r_{32} & \alpha_v r_{23} + v_0 r_{33} & \alpha_v t_2 + v_0 t_3 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

Applying the Direct Linear Transform (DLT) algorithm

The idea of the DLT is to rewrite the perspective projection equation as a **homogeneous linear equation** and solve it by standard methods. Let's write the perspective equation for a generic 3D-2D point correspondence:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K\left[R \mid T\right] \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad \Rightarrow$$

$$\Rightarrow \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

$$\Rightarrow \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \boxed{\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix}} \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

What are the assumptions behind this this substitution?

Applying the Direct Linear Transform (DLT) algorithm

$$\underbrace{\begin{pmatrix} X_w^1 & Y_w^1 & Z_w^1 & 1 & 0 & 0 & 0 & 0 & -u_1 X_w^1 & -u_1 Y_w^1 & -u_1 Z_w^1 & -u_1 \\ 0 & 0 & 0 & 0 & X_w^1 & Y_w^1 & Z_w^1 & 1 & -v_1 X_w^1 & -v_1 Y_w^1 & -v_1 Z_w^1 & -v_1 \\ & & & & & & & \vdots & & & & \\ X_w^n & Y_w^n & Z_w^n & 1 & 0 & 0 & 0 & 0 & -u_n X_w^n & -u_n Y_w^n & -u_n Z_w^n & -u_n \\ 0 & 0 & 0 & 0 & X_w^n & Y_w^n & Z_w^n & 1 & -v_n X_w^n & -v_n Y_w^n & -v_n Z_w^n & -v_n \end{pmatrix}}_{\text{Q (this matrix is } \textbf{known}\text{)}} \underbrace{\begin{pmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{pmatrix}}_{\text{M (this matrix is } \textbf{unknown}\text{)}} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \Rightarrow \quad \mathrm{Q} \cdot \mathrm{M} = 0$$

Applying the Direct Linear Transform (DLT) algorithm

$$Q \cdot M = 0$$

Minimal solution

- $Q_{(2n\times12)}$ should have rank 11 to have a unique (up to a scale) *non-zero* solution M
- Because each 3D-to-2D point correspondence provides 2 independent equations, then $5+\frac{1}{2}$ point correspondences are needed (in practice **6 point** correspondences!)

Over-determined solution

- For $n \geq 6$ points, a solution is the **Least Square solution**, which minimizes the sum of squared residuals, $||QM||^2$, subject to the constraint $||M||^2 = 1$. It can be solved through Singular Value Decomposition (SVD). The solution is the eigenvector corresponding to the smallest eigenvalue of the matrix Q^TQ (because it is the unit vector x that minimizes $||Qx||^2 = x^TQ^TQx$.
- Matlab instructions:
  - `[U,S,V] = SVD(Q);`
  - `M = V(:,12);`

Applying the Direct Linear Transform (DLT) algorithm

- Once we have determined M, we can recover the intrinsic and extrinsic parameters by remembering that:

$$\mathrm{M} = \mathrm{K}(\mathrm{R} \mid \mathrm{T})$$

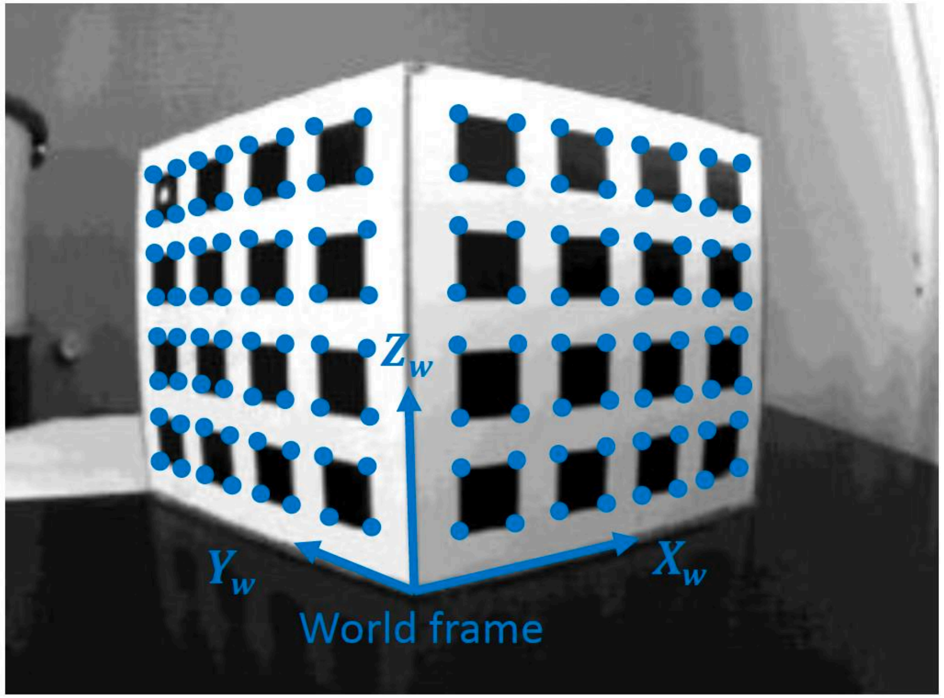
$$\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

Considering the first three columns of M, it is equal to K R,
the product of an upper triangular matrix and an orthogonal matrix

We can use the QR decomposition from linear algebra

Example of Tsai's Calibration Results

Recommendation: use many **more than 6 points** (ideally more than 20) **and non coplanar**



Corners can be detected with accuracy < 0.1 pixels (see Lecture 5)

α_u	α_u/α_v	K_{12}	u_0	v_0	**Average Reprojection error**
1673.3	1.0063	1.39	379.96	305.78	0.365

Why is this ratio not 1?

What is this?

What is this?

How can we estimate the lens distortion parameters?

How can we enforce $\alpha_u = \alpha_v$ and $K_{12} = 0$?

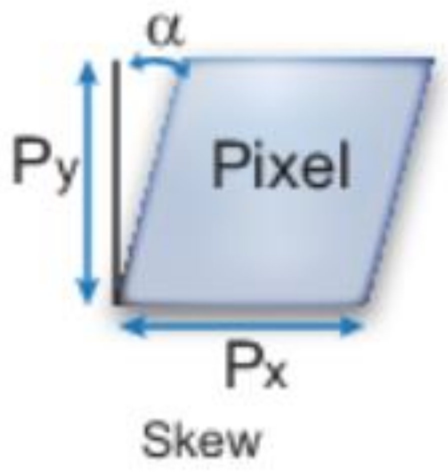
Intrinsic Parameters

(figures from https://www.mathworks.com/help/vision/ug/camera-calibration.html)

$$\begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

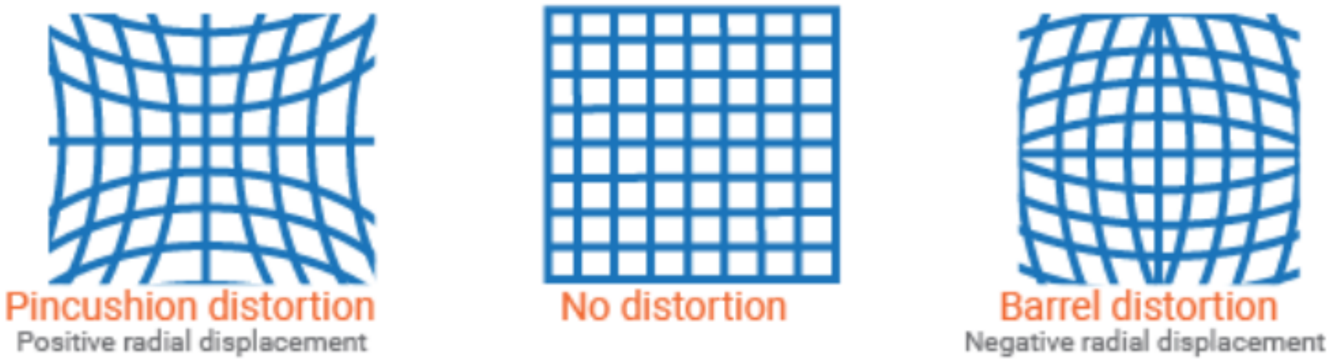
$[c_x \quad c_y]$ – Optical center (the principal point), in pixels.
$(f_x, \quad f_y)$ – Focal length in pixels. $f_x = F/p_x$ $f_y = F/p_y$ F – Focal length in world units, typically expressed in millimeters. $(p_x, \quad p_y)$ – Size of the pixel in world units.
s – Skew coefficient, which is non-zero if the image axes are not perpendicular. $s = f_x \tan \alpha$

The pixel skew is defined as:



Non-linear Lens Distortion

(figures from https://www.mathworks.com/help/vision/ug/camera-calibration.html)



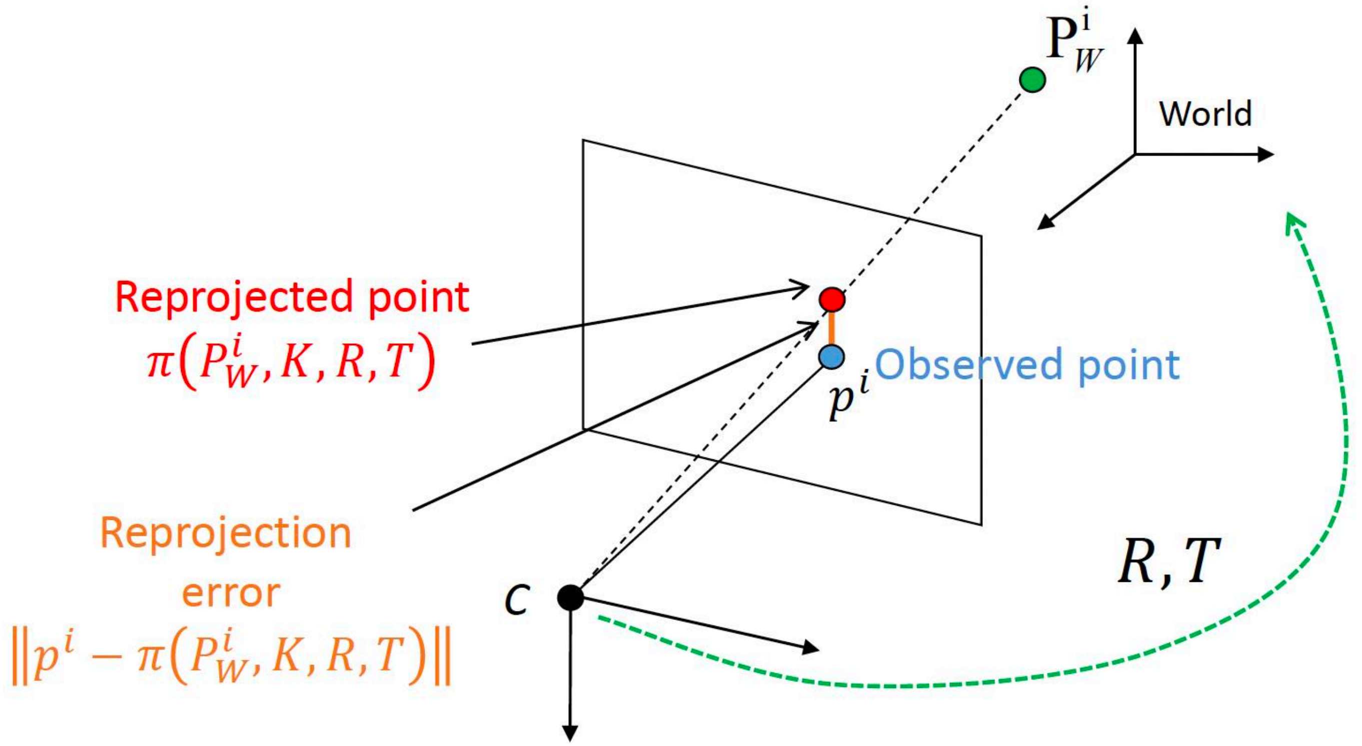
The radial distortion coefficients model this type of distortion. The distorted points are denoted as ($x_{\text{distorted}}$, $y_{\text{distorted}}$):

$$x_{\text{distorted}} = x(1 + k_1*r^2 + k_2*r^4 + k_3*r^6)$$

$$y_{\text{distorted}} = y(1 + k_1*r^2 + k_2*r^4 + k_3*r^6)$$

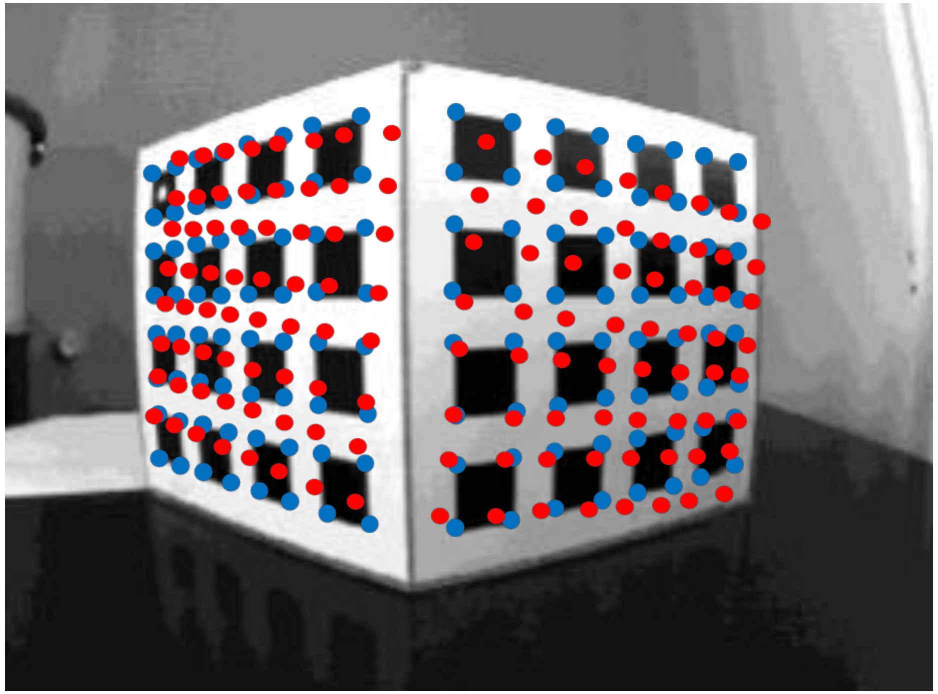
Reprojection Error

- The reprojection error is the **Euclidean distance** (in pixels) between an **observed image point** and the **corresponding** 3D point **reprojected** onto the camera frame.
- The reprojection error gives us a **quantitative measure of the accuracy** of the calibration (**ideally it should be zero**).



Reprojection Error

- The reprojection error can be used to assess the quality of the camera calibration
- What reprojection error is acceptable?
- What are the sources of the reprojection error?
- How can we further improve the calibration parameters?



Control points (observed points)

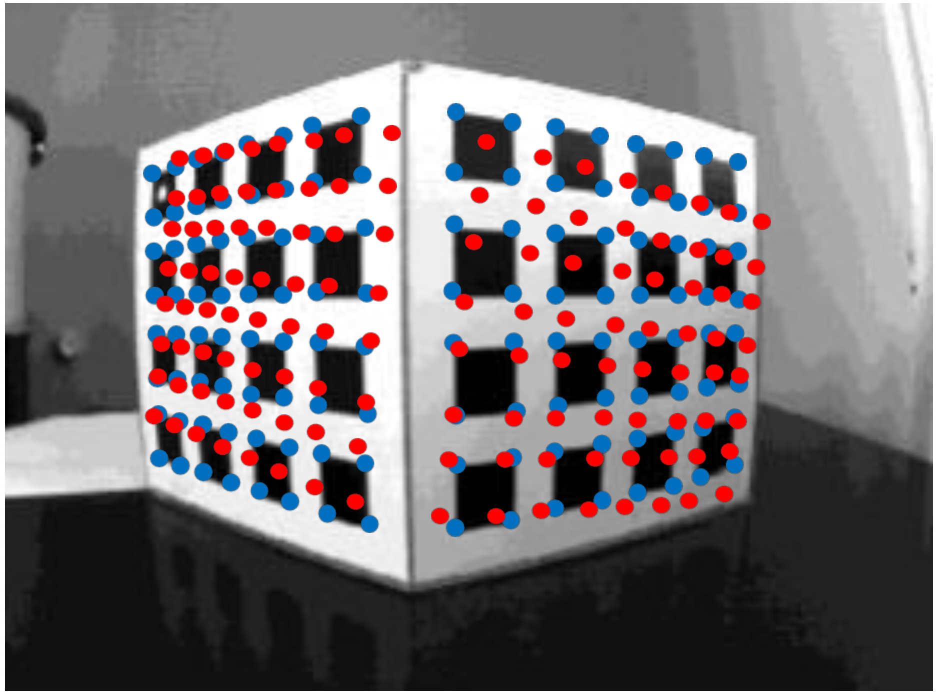
Reprojected points $\pi\left(P_W^i, K, R, T\right)$

Non-Linear Calibration Refinement

- The calibration parameters K, R, T determined by the DLT can be refined by minimizing the following cost:

$$K, R, T, lens\ distortion = argmin_{K,R,T,lens} \sum_{i=1}^{n} \left\| p^i - \pi\left(P_W^i, K, R, T\right) \right\|^2$$

- This time we also include the **lens distortion** (can be set to 0 for initialization)
- Can be minimized using **Levenberg–Marquardt** (more robust than Gauss-Newton to local minima)



Control points (observed points)

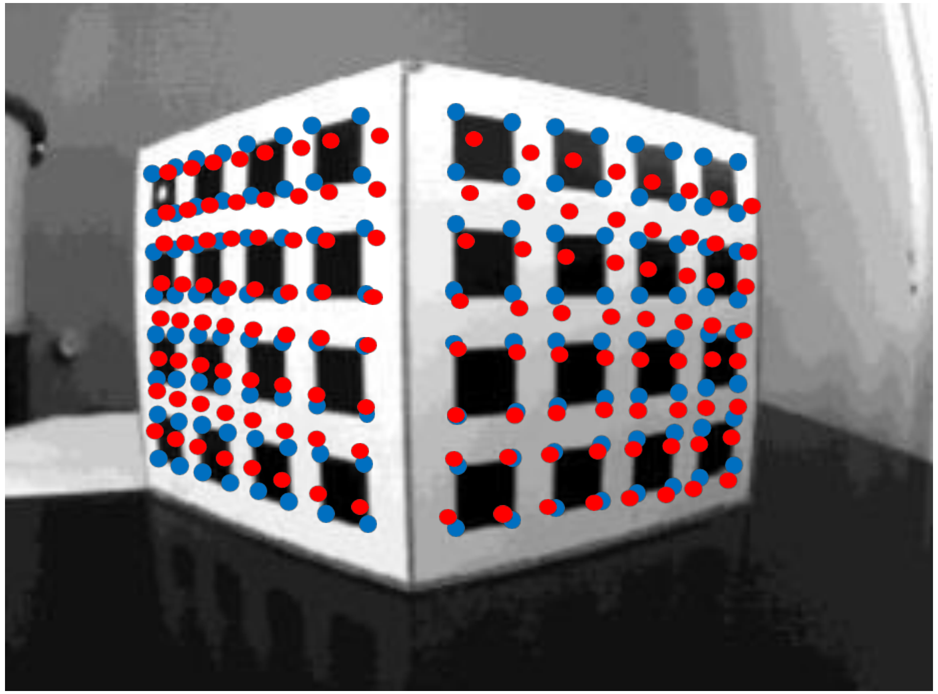
Reprojected points $\pi\left(P_W^i, K, R, T\right)$

Non-Linear Calibration Refinement

- The calibration parameters K, R, T determined by the DLT can be refined by minimizing the following cost:

$$K, R, T, lens\ distortion = argmin_{K,R,T,lens} \sum_{i=1}^{n} \left\| p^i - \pi\left(P_W^i, K, R, T\right) \right\|^2$$

- This time we also include the **lens distortion** (can be set to 0 for initialization)
- Can be minimized using **Levenberg–Marquardt** (more robust than Gauss-Newton to local minima)



Control points (observed points)

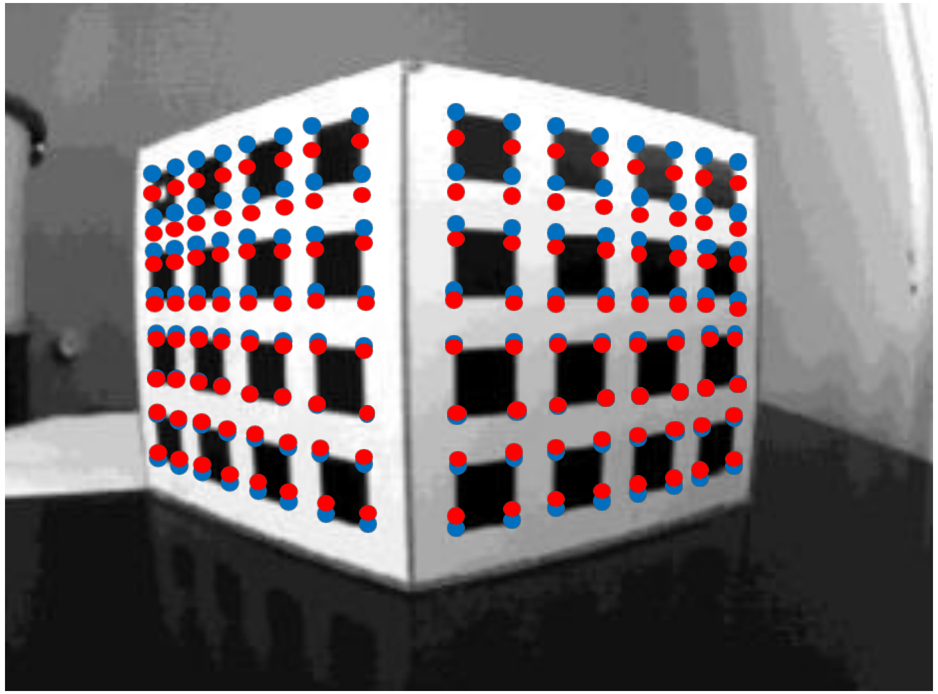
Reprojected points $\pi\left(P_W^i, K, R, T\right)$

Non-Linear Calibration Refinement

- The calibration parameters K, R, T determined by the DLT can be refined by minimizing the following cost:

$$K, R, T, lens\ distortion = \\ argmin_{K,R,T,lens} \sum_{i=1}^{n} \left\| p^i - \pi\left(P_W^i, K, R, T\right) \right\|^2$$

- This time we also include the **lens distortion** (can be set to 0 for initialization)
- Can be minimized using **Levenberg–Marquardt** (more robust than Gauss-Newton to local minima)



Control points (observed points)

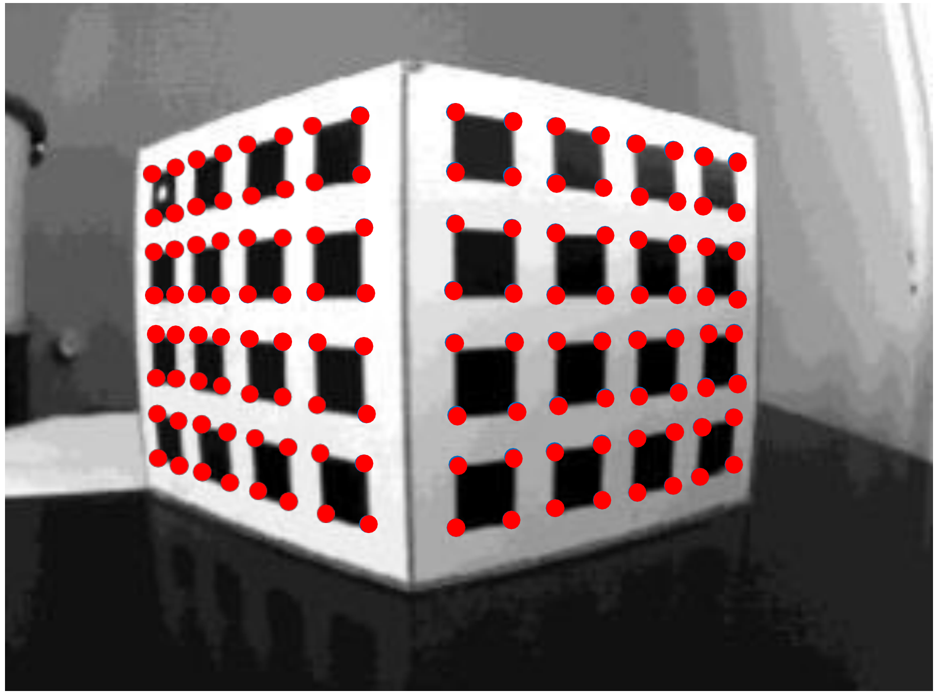
Reprojected points $\pi\left(P_W^i, K, R, T\right)$

Non-Linear Calibration Refinement

- The calibration parameters K, R, T determined by the DLT can be refined by minimizing the following cost:

$$K, R, T, lens\ distortion = argmin_{K,R,T,lens} \sum_{i=1}^{n} \left\| p^i - \pi\left(P_W^i, K, R, T\right) \right\|^2$$

- This time we also include the **lens distortion** (can be set to 0 for initialization)
- Can be minimized using **Levenberg–Marquardt** (more robust than Gauss-Newton to local minima)

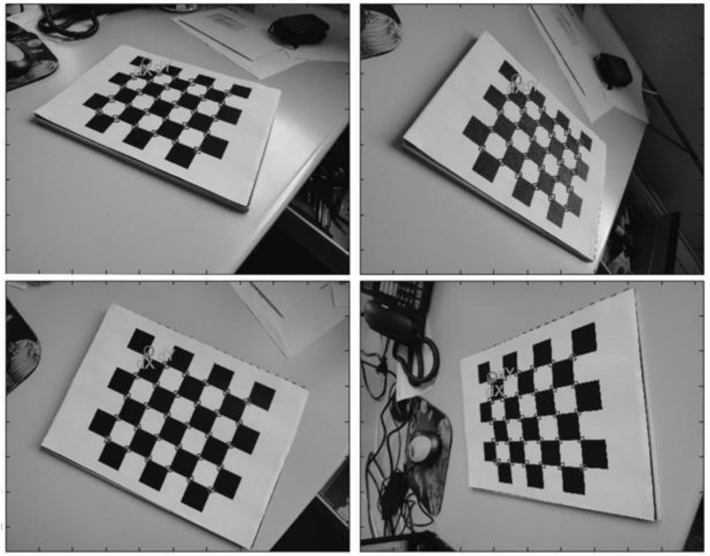


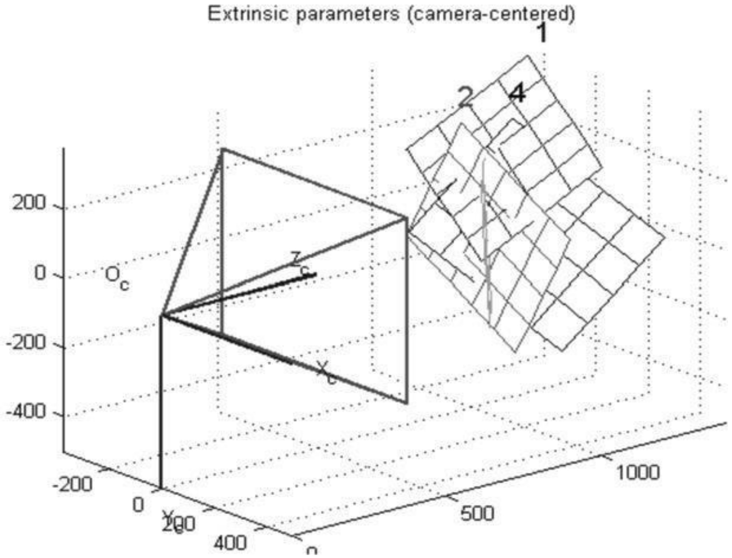
Control points (observed points)

Reprojected points $\pi\left(P_W^i, K, R, T\right)$

Zhang's Algorithm: Calibration from Planar Grids

- **Tsai's calibration** requires that the world's 3D points are non-coplanar, which is **not very practical**
- **Today's camera calibration toolboxes** (Matlab, OpenCV) use **multiple views** of a **planar grid** (e.g., a checker board)
- They are based on a method developed in 2000 by Zhang (Microsoft Research)

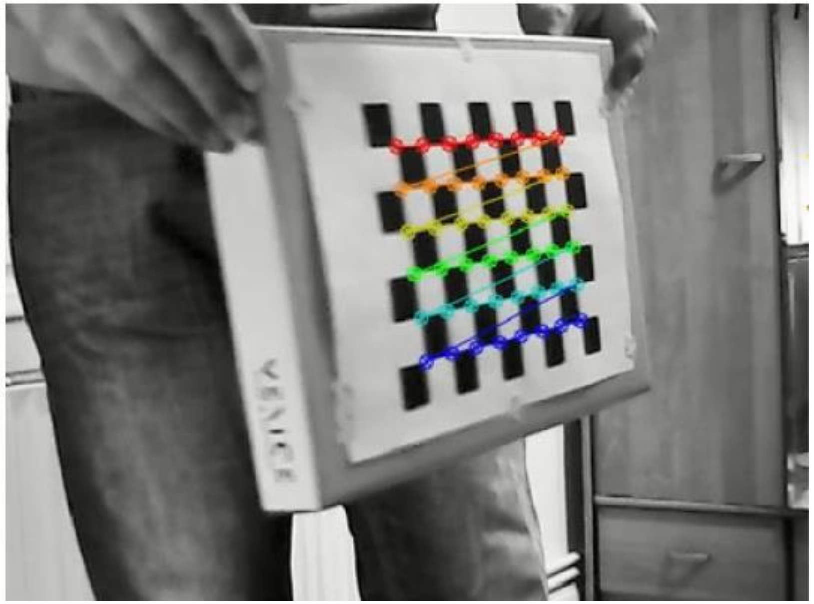




Zhang, A flexible new technique for camera calibration, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000. PDF.

Zhang's Algorithm: Calibration from Planar Grids

- **Tsai's calibration** requires that the world's 3D points are non-coplanar, which is **not very practical**
- **Today's camera calibration toolboxes** (Matlab, OpenCV) use **multiple views** of a **planar grid** (e.g., a checker board)
- They are based on a method developed in 2000 by Zhang (Microsoft Research)



Zhang, A flexible new technique for camera calibration, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000. PDF.

Applying the Direct Linear Transform (DLT) algorithm

As in Tsai's method, we start by writing the perspective projection equation (again, we neglect the radial distortion). However, in **Zhang's method the points are all coplanar**, i.e., $\boldsymbol{Z_w = 0}$, and thus we can write:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K\left[R \mid T\right] \cdot \begin{bmatrix} X_w \\ Y_w \\ 0 \\ 1 \end{bmatrix} \quad \Rightarrow$$

$$\Rightarrow \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ 0 \\ 1 \end{bmatrix}$$

$$\Rightarrow \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix}$$

Applying the Direct Linear Transform (DLT) algorithm

As in Tsai's method, we start by writing the perspective projection equation (again, we neglect the radial distortion). However, in **Zhang's method the points are all coplanar**, i.e., $\boldsymbol{Z_w = 0}$, and thus we can write:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[R \mid T] \cdot \begin{bmatrix} X_w \\ Y_w \\ 0 \\ 1 \end{bmatrix} \quad \Rightarrow$$

$$\Rightarrow \quad \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ 0 \\ 1 \end{bmatrix}$$

$$\Rightarrow \quad \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix}$$

Applying the Direct Linear Transform (DLT) algorithm

$$\Rightarrow \quad \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix}$$

$$\Rightarrow \quad \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = H \cdot \begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix}$$

This matrix is called Homography

$$\Rightarrow \quad \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} h_1^{\mathrm{T}} \\ h_2^{\mathrm{T}} \\ h_3^{\mathrm{T}} \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix}$$

where h_i^{T} is the i-*th* row of H

Applying the Direct Linear Transform (DLT) algorithm

$$\Rightarrow \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} h_1^T \\ h_2^T \\ h_3^T \end{bmatrix} \cdot \underbrace{\begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix}}_{\rightarrow P}$$

Conversion back from homogeneous coordinates to pixel coordinates leads to:

$$u = \frac{\lambda u}{\lambda} = \frac{h_1^T \cdot P}{h_3^T \cdot P} \qquad \Rightarrow \qquad (h_1^T - u_i h_3^T) \cdot P_i = 0$$

$$v = \frac{\lambda v}{\lambda} = \frac{h_2^T \cdot P}{h_3^T \cdot P} \qquad \qquad (h_2^T - v_i h_3^T) \cdot P_i = 0$$

Applying the Direct Linear Transform (DLT) algorithm

- By re-arranging the terms, we obtain:

$$\begin{aligned}(h_1^{\mathrm{T}} - u_i h_3^{\mathrm{T}}) \cdot P_i = 0 \\ (h_2^{\mathrm{T}} - v_i h_3^{\mathrm{T}}) \cdot P_i = 0\end{aligned} \quad \Rightarrow \quad \begin{aligned} P_i^{\mathrm{T}} \cdot h_1 + 0 \cdot h_2^{\mathrm{T}} - u_i P_i^{\mathrm{T}} \cdot h_3^{\mathrm{T}} = 0 \\ 0 \cdot h_1^{\mathrm{T}} + P_i^{\mathrm{T}} \cdot h_2^{\mathrm{T}} - v_i P_i^{\mathrm{T}} \cdot h_3^{\mathrm{T}} = 0 \end{aligned} \quad \Rightarrow \begin{pmatrix} P_i^{\mathrm{T}} & 0^{\mathrm{T}} & -u_1 P_i^{\mathrm{T}} \\ 0^{\mathrm{T}} & P_i^{\mathrm{T}} & -v_1 P_i^{\mathrm{T}} \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

- For n points (from a **single view**), we can stack all these equations into a big matrix:

$$\underbrace{\begin{pmatrix} P_1^{\mathrm{T}} & 0^{\mathrm{T}} & -u_1 P_1^{\mathrm{T}} \\ 0^{\mathrm{T}} & P_1^{\mathrm{T}} & -v_1 P_1^{\mathrm{T}} \\ \cdots & \cdots & \cdots \\ P_n^{\mathrm{T}} & 0^{\mathrm{T}} & -u_n P_n^{\mathrm{T}} \\ 0^{\mathrm{T}} & P_n^{\mathrm{T}} & -v_n P_n^{\mathrm{T}} \end{pmatrix}}_{\text{Q (this matrix is known)}} \underbrace{\begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix}}_{\text{H (this matrix is unknown)}} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \quad \Rightarrow \quad \mathrm{Q} \cdot \mathrm{H} = 0$$

Q (this matrix is **known**) H (this matrix is **unknown**)

Applying the Direct Linear Transform (DLT) algorithm

$$Q \cdot H = 0$$

Minimal solution

- $Q_{(2n\times 9)}$ should have rank 8 to have a unique (up to a scale) non-trivial solution H
- Each point correspondence provides 2 independent equations
- Thus, a minimum of **4 non-collinear points** is required

Over-determined solution

- $n \geq 4$ points
- It can be solved through Singular Value Decomposition (SVD) (same considerations as before)

How to recover K, R, T

- H can be decomposed by recalling that:

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix}$$

- Differently from Tsai's, the decomposition of H into K, R, T requires **at least two views**
- In practice **the more views the better**, e.g., 20-50 views spanning the entire field of view of the camera for the best calibration results
- Notice that now **each view j has a different homography** H^j (and so a different R^j and T^j). However, ***K* is the same for all views**:

$$\begin{bmatrix} h_{11}^j & h_{12}^j & h_{13}^j \\ h_{21}^j & h_{22}^j & h_{23}^j \\ h_{31}^j & h_{33}^j & h_{33}^j \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11}^j & r_{12}^j & t_1^j \\ r_{21}^j & r_{22}^j & t_2^j \\ r_{31}^j & r_{32}^j & t_3^j \end{bmatrix}$$