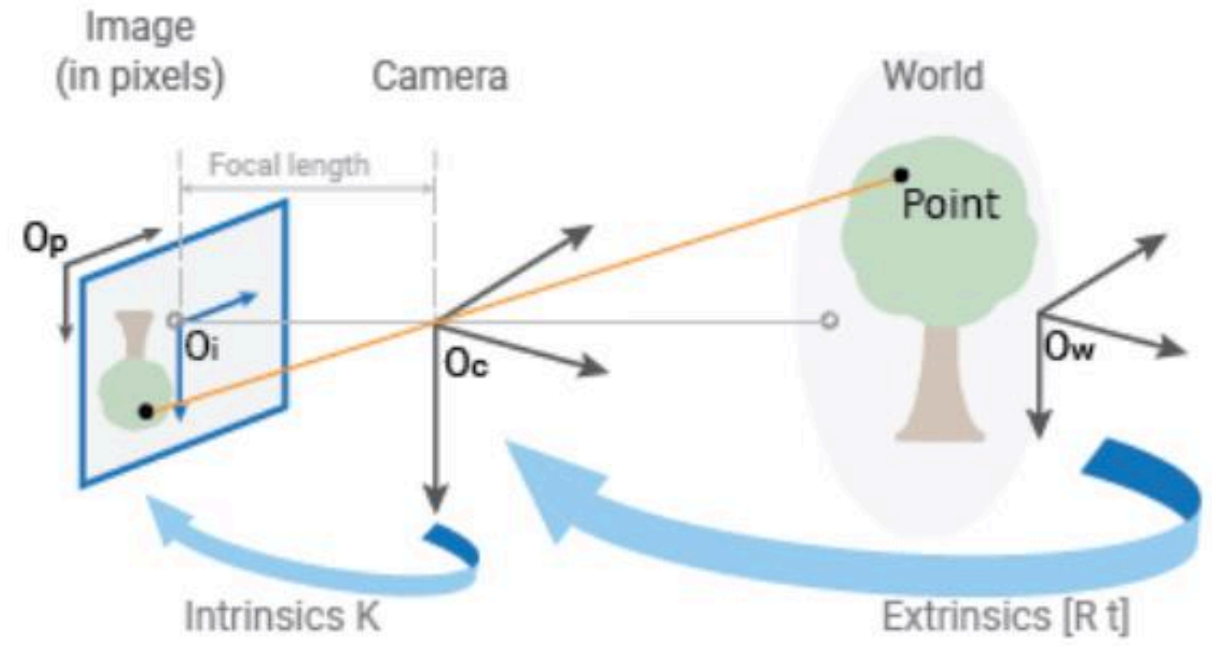
Camera Calibration – DLT, Zhang, PnP

Lecture based on material from Davide Scaramuzza’s course

Camera Calibration

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



$$w\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = P\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Scale factor

Image points

World points

$$P = K\begin{bmatrix} R & t \end{bmatrix}$$

Camera matrix

Intrinsics matrix

Extrinsics
Rotation and Translation

Vision Algorithms for Mobile Robotics

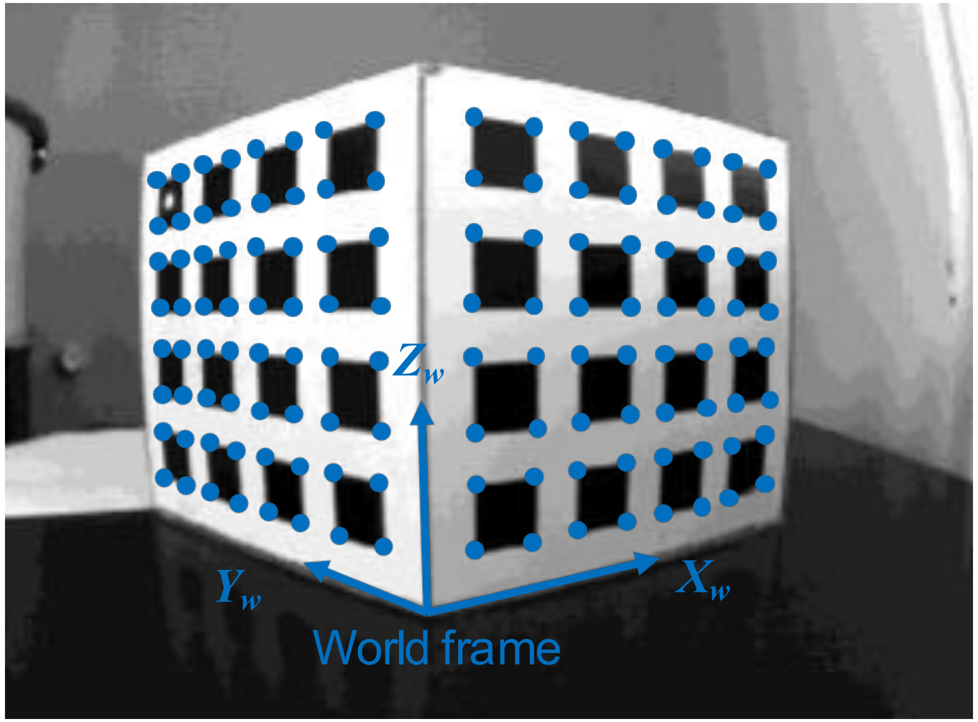
Lecture 03
Camera Calibration

Davide Scaramuzza

http://rpg.ifi.uzh.ch

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:

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

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

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

What are the assumptions behind this this substitution?

Applying the Direct Linear Transform (DLT) algorithm

$$
\underbrace{\begin{pmatrix}
X_w^1 & Y_w^1 & Z_w^1 & 1 & 0 & 0 & 0 & 0 & -u_1X_w^1 & -u_1Y_w^1 & -u_1Z_w^1 & -u_1 \\
0 & 0 & 0 & 0 & X_w^1 & Y_w^1 & Z_w^1 & 1 & -v_1X_w^1 & -v_1Y_w^1 & -v_1Z_w^1 & -v_1 \\
 & & & & & & & \vdots & & & & \\
X_w^n & Y_w^n & Z_w^n & 1 & 0 & 0 & 0 & 0 & -u_nX_w^n & -u_nY_w^n & -u_nZ_w^n & -u_n \\
0 & 0 & 0 & 0 & X_w^n & Y_w^n & Z_w^n & 1 & -v_nX_w^n & -v_nY_w^n & -v_nZ_w^n & -v_n
\end{pmatrix}}_{\text{Q (this matrix is \textbf{known})}}
\underbrace{\begin{pmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{pmatrix}}_{\text{M (this matrix is \textbf{unknown})}}
= \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}
\Rightarrow \quad \mathrm{Q} \cdot \mathrm{M} = 0
$$

Applying the Direct Linear Transform (DLT) algorithm

$$Q \cdot M = 0$$

Minimal solution

- $Q_{(2n\times 12)}$ 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:

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

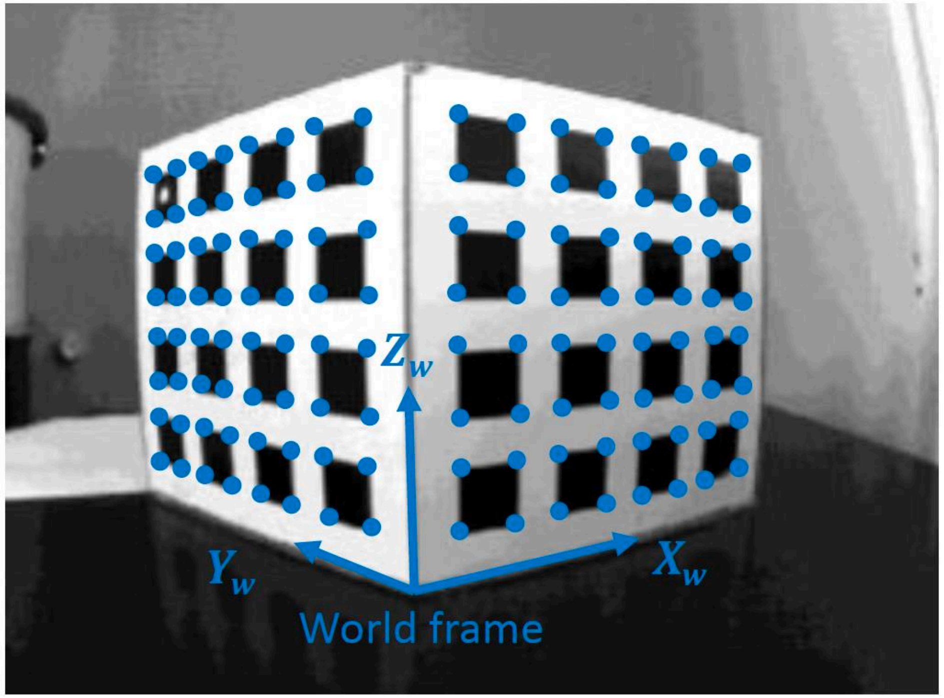
$$\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23}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

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**



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?

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

How can we estimate the lens distortion parameters?
How can we enforce $\alpha_u = \alpha_v$ and $K_{12} = 0$?

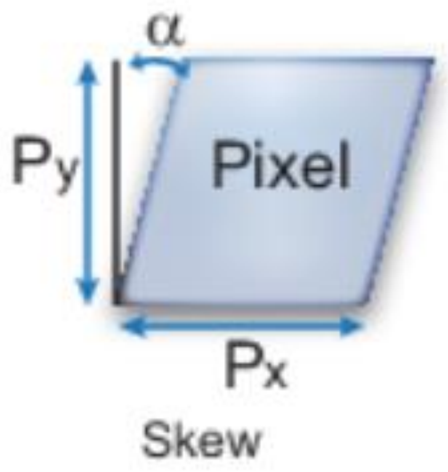
Intrinsic Parameters

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

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

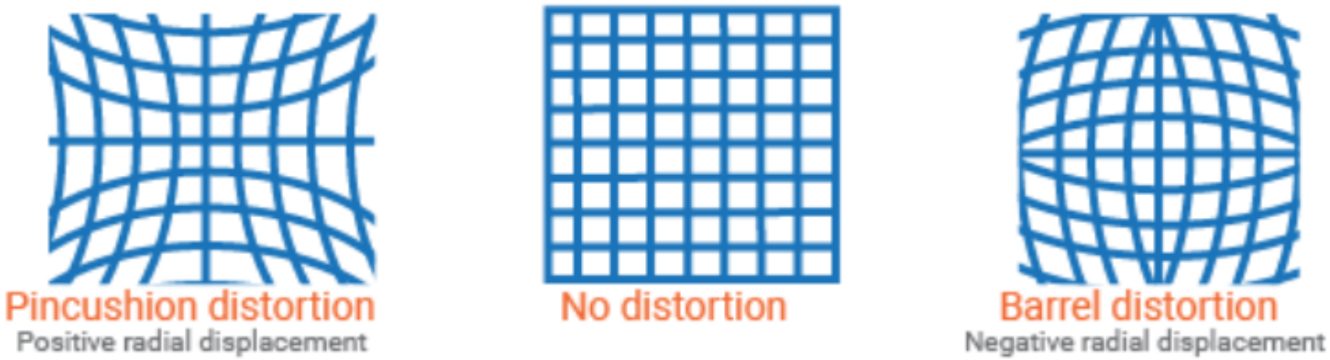
$[c_x \quad c_y]$ – Optical center (the principal point), in pixels.
$(f_x, \quad f_y)$ – Focal length in pixels. $f_x = F/p_x$ $f_y = F/p_y$ F – Focal length in world units, typically expressed in millimeters. $(p_x, \quad p_y)$ – Size of the pixel in world units.
s – Skew coefficient, which is non-zero 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)



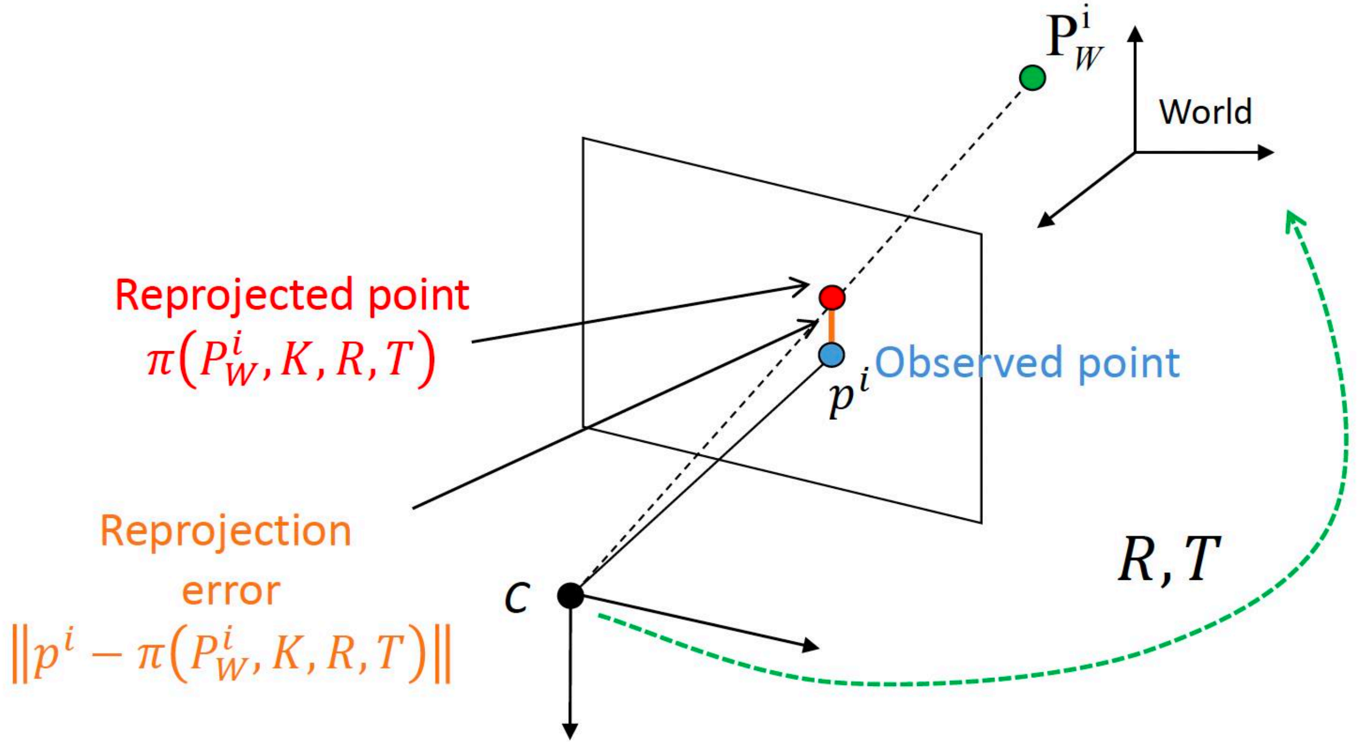
The radial distortion coefficients model this type of distortion. The distorted points are denoted as ($x_{\text{distorted}}$, $y_{\text{distorted}}$):

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

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

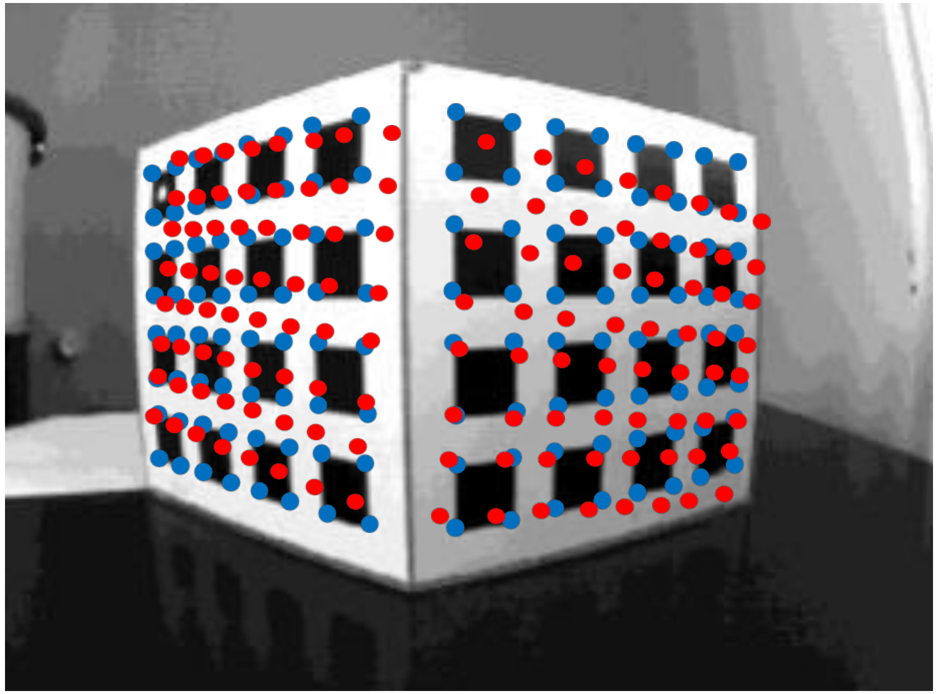
Reprojection Error

- 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)

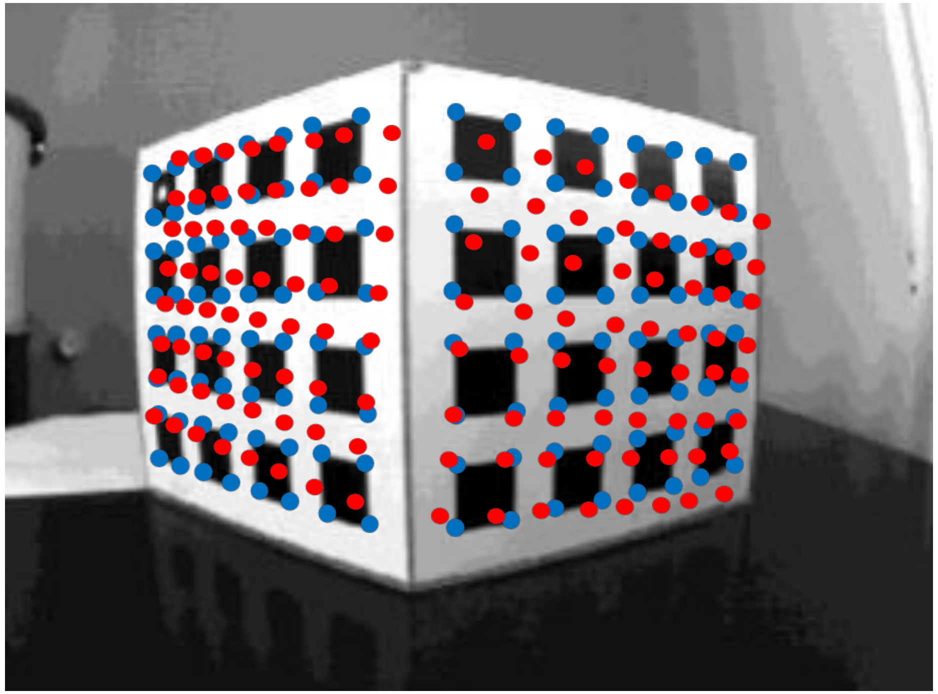
Reprojected points $\pi\left(P_W^i, K, R, T\right)$

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)

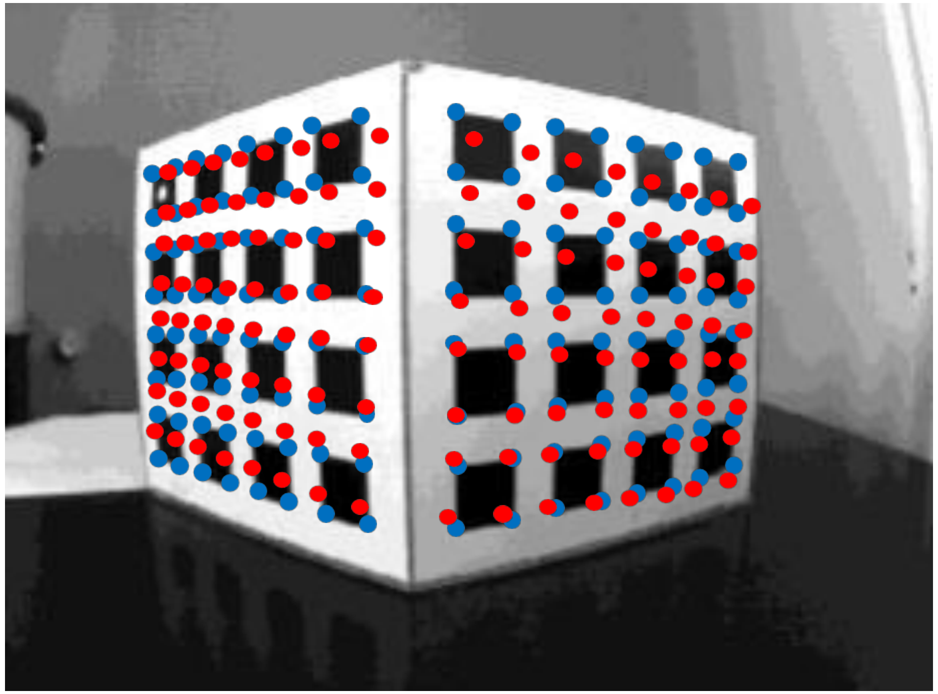
Reprojected points $\pi\left(P_W^i, K, R, T\right)$

Non-Linear Calibration Refinement

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

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

- 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)

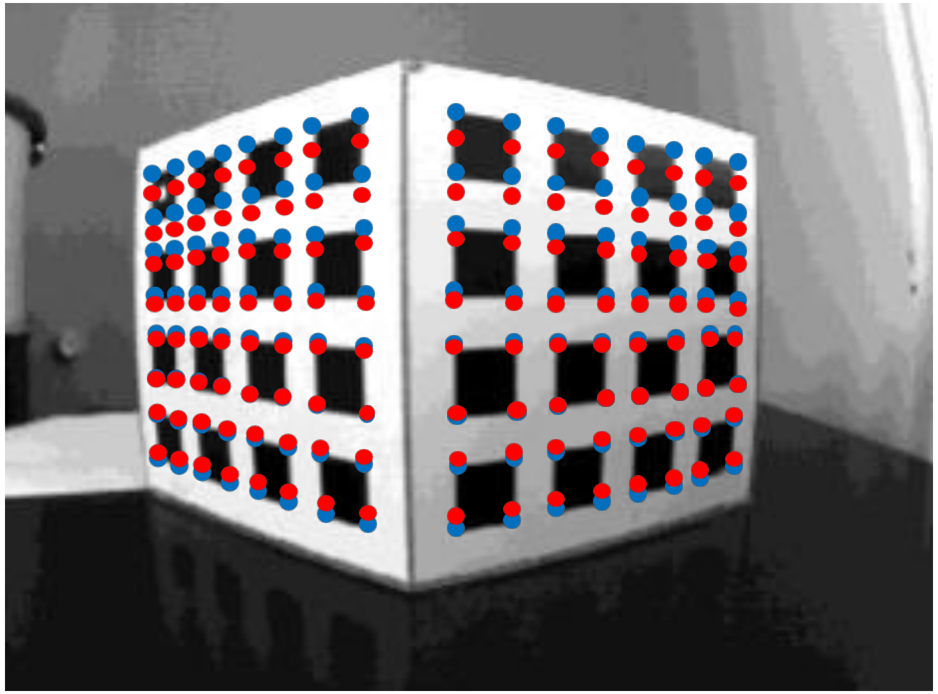
Reprojected points $\pi\left(P_W^i, K, R, T\right)$

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)

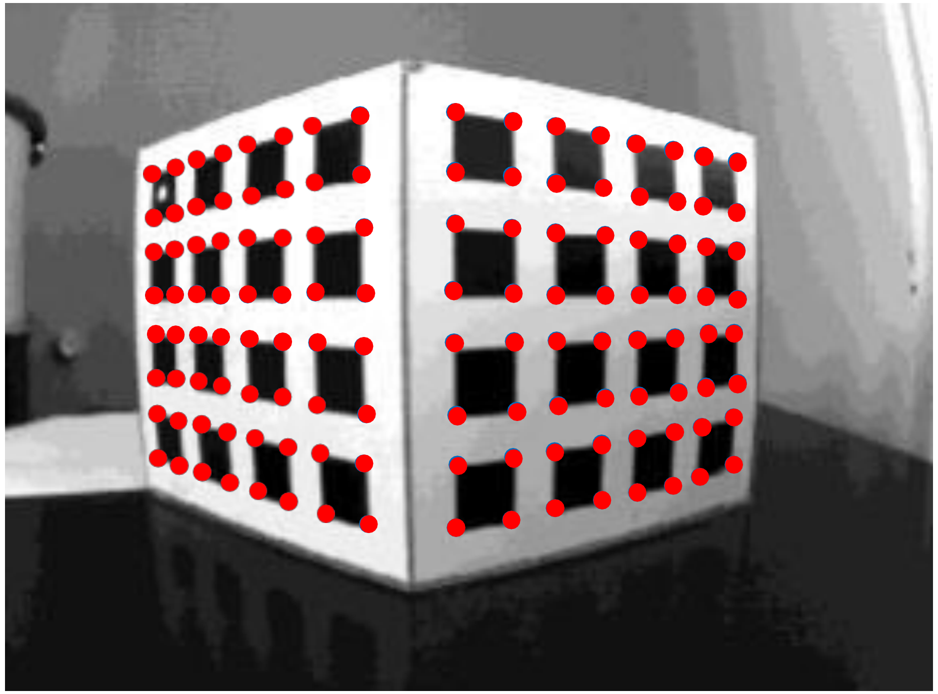
Reprojected points $\pi\left(P_W^i, K, R, T\right)$

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)

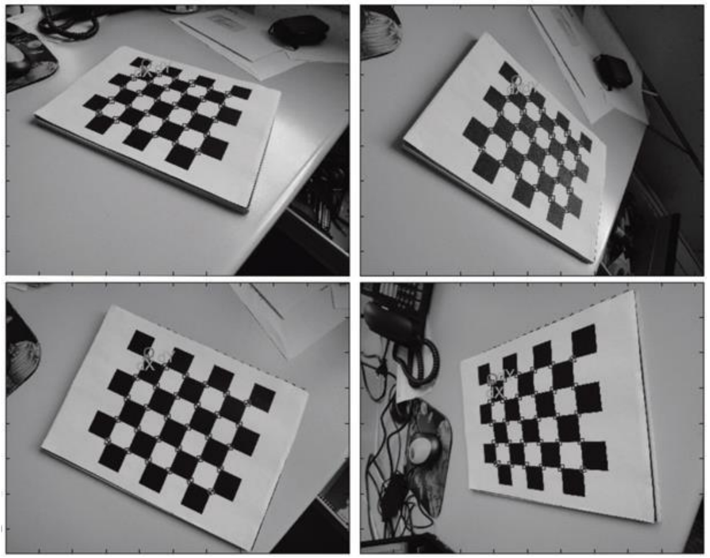


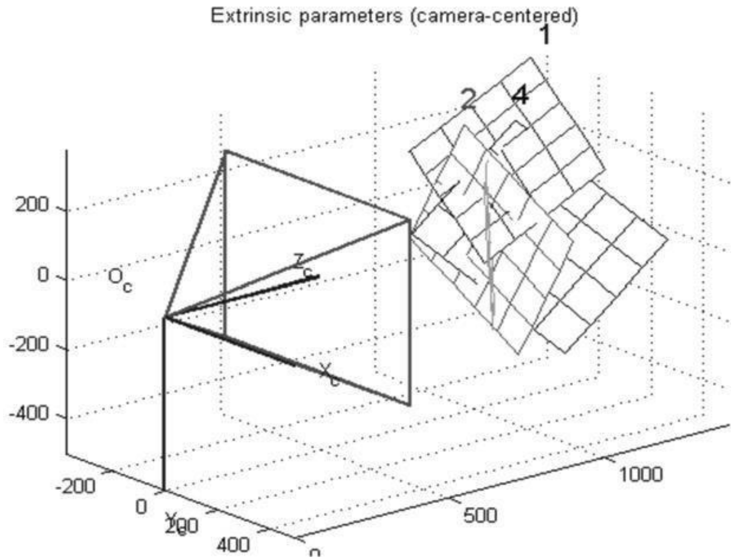
Control points (observed points)

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)

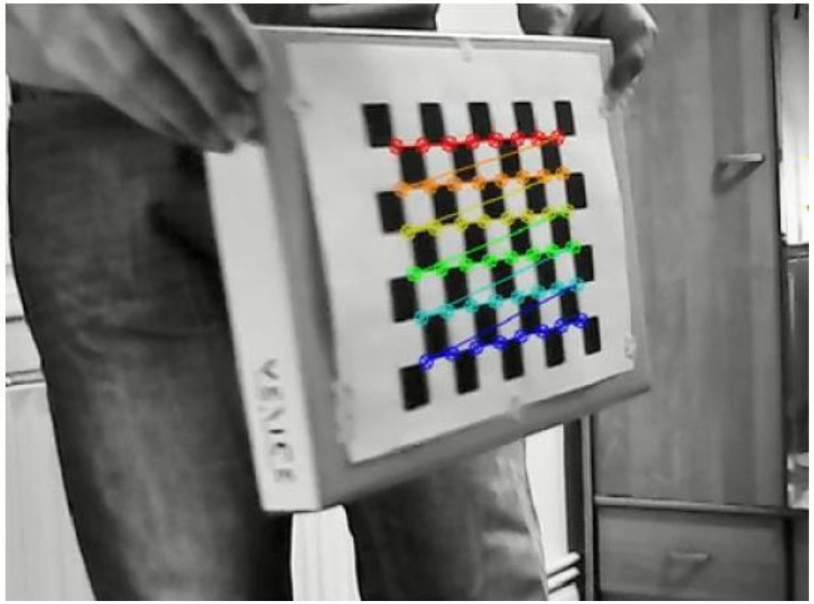




Zhang, A flexible new technique for camera calibration, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000. PDF.

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:

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

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

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

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:

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

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

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

Applying the Direct Linear Transform (DLT) algorithm

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

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

This matrix is called Homography

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

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

Applying the Direct Linear Transform (DLT) algorithm

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

Conversion back from homogeneous coordinates to pixel coordinates leads to:

$$u = \frac{\lambda u}{\lambda} = \frac{h_1^{\mathrm{T}} \cdot P}{h_3^{\mathrm{T}} \cdot P}$$

$$v = \frac{\lambda v}{\lambda} = \frac{h_2^{\mathrm{T}} \cdot P}{h_3^{\mathrm{T}} \cdot P}$$

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

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:

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

Applying the Direct Linear Transform (DLT) algorithm

$$Q \cdot H = 0$$

Minimal solution

- $Q_{(2n\times9)}$ 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:

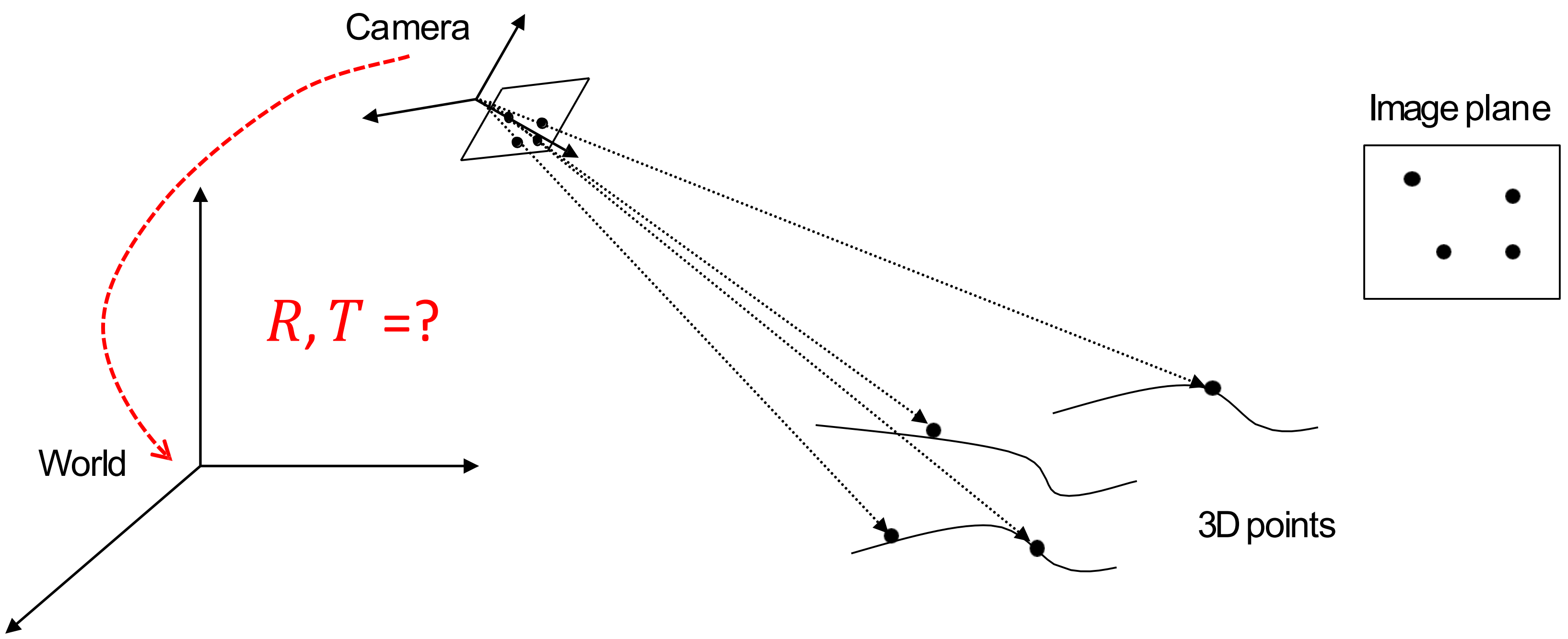
$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix}$$

- 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**:

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

Camera Localization (or Perspective from n Points: PnP)

- This is the problem of determining the **6DoF pose of a camera** (position and orientation) with respect to the world frame **from a set of 3D-2D point correspondences**.
- It assumes the **camera** to be **already calibrated**
- The **DLT can be used** to solve this problem **but is suboptimal**. We want to study **algebraic solutions** to the problem.



3 Points (P3P problem)

- **3 Points** (non collinear):
  - up to 4 solution

$$s_1^2 = L_B^2 + L_A^2 - 2L_B L_A \cos\theta_{AB}$$

$$s_2^2 = L_A^2 + L_C^2 - 2L_A L_C \cos\theta_{AC}$$

$$s_3^2 = L_B^2 + L_C^2 - 2L_B L_C \cos\theta_{BC}$$

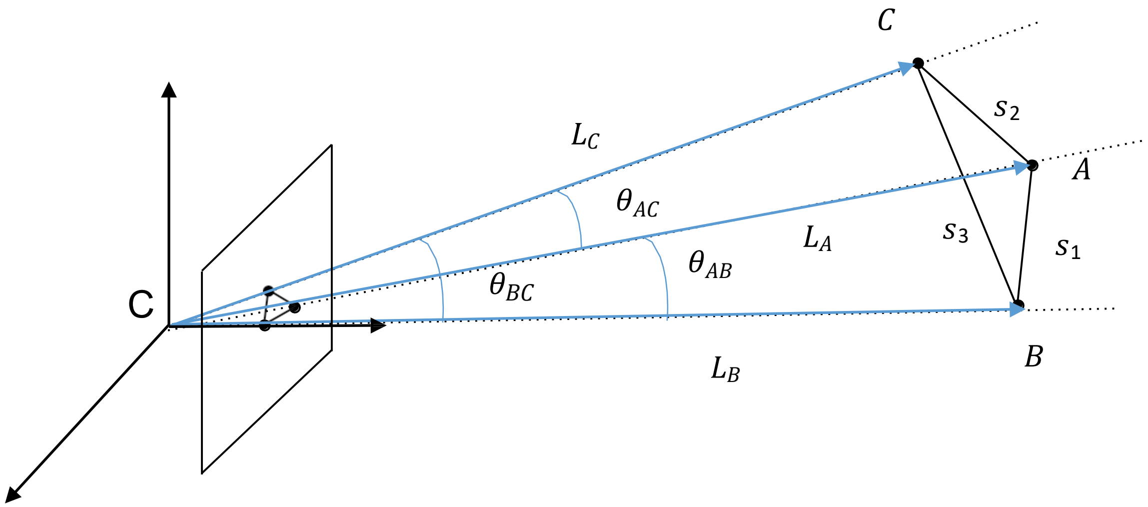
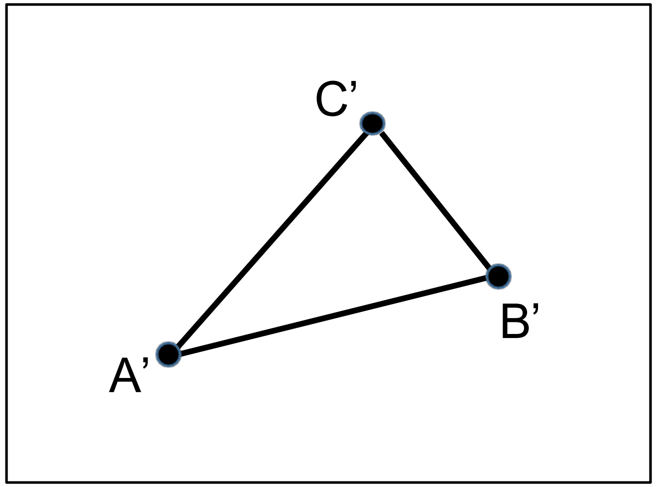


Image plane



Algebraic Approach: reduce to 4th order equation

$$s_1^2 = L_B^2 + L_A^2 - 2L_B L_A \cos\theta_{AB}$$

$$s_2^2 = L_A^2 + L_C^2 - 2L_A L_C \cos\theta_{AC}$$

$$s_3^2 = L_B^2 + L_C^2 - 2L_B L_C \cos\theta_{BC}$$

- It is known that n **independent polynomial equations**, in n **unknowns**, can have no more **solutions** than the **product of their respective degrees**. Thus, the system can have a maximum of **8** solutions. However, because every term in the system is either a constant or of second degree, for every real positive solution **there is a negative solution**.
- Thus, with 3 points, there are at most **4 valid (positive) solutions**.

M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Graphics and Image Processing, 1981. PDF.

Algebraic Approach: reduce to 4th order equation

$$s_1^2 = L_B^2 + L_A^2 - 2L_B L_A \cos\theta_{AB}$$

$$s_2^2 = L_A^2 + L_C^2 - 2L_A L_C \cos\theta_{AC}$$

$$s_3^2 = L_B^2 + L_C^2 - 2L_B L_C \cos\theta_{BC}$$

- By defining $\boldsymbol{x} = \boldsymbol{L_B}/\boldsymbol{L_A}$, it can be shown that the system can be reduced to a 4th order equation:

$$G_0 + G_1 x + G_2 x^2 + G_3 x^3 + G_4 x^4 = 0$$

How can we disambiguate the 4 solutions? How do we determine R and T?

- A 4th point can be used to disambiguate the solutions. A classification of the four solutions and the determination of R and T from the point distances was given Gao's algorithm, implemented in OpenCV (*solvePnP_P3P*)

Gao, Hou, Tang, Cheng. Complete Solution Classification for the Perspective-Three-Point Problem. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2003. PDF.

PnP problem: Recap

Calibrated camera (i.e., instrinc parameters are known)	**Uncalibrated camera** (i.e., intrinsic parameters unknown)
Either DLT or EPnP can be used	**Only DLT** can be used

EPnP: minimum number of points: **3 (P3P) +1** for disambiguation
DLT: Minimum number of points: **4 if coplanar, 6 if non-coplanar**

The output of both DLT and EPnP can be refined via **non-linear optimization** by minimizing the sum of squared reprojection errors

Some history…

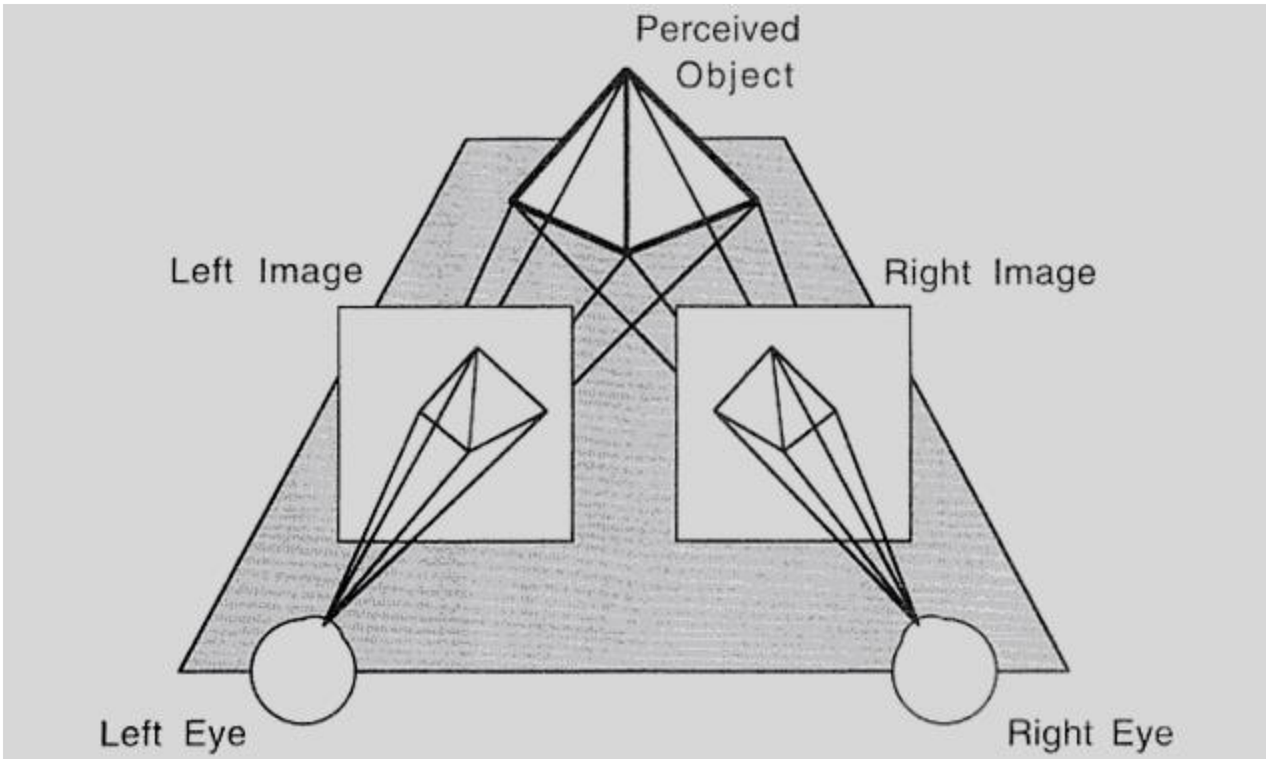
- In 1851, the French inventor Aimé Laussedat saw the possibility of using the newly invented camera in mapping.
- In 1867, Prussian architect Albrecht Medenbauer coined the name photogrammetry in his article “Die Photometrophie.”
- Substantial contributions were made by Sebastian Finsterwalder and by Erwin Kruppa, who established the structure-from-motion theorem in 1913.
- In the period preceding World War I and World War II, aerial photogrammetry found widespread use.
- Computer vision researchers independently rediscovered many of these results in the 1970s and 1980s; by the 1990s the classical literature had been “found”.
- The terminology is slightly different from that used in computer vision e.g. finding intrinsic parameters is the “interior orientation” problem, extrinsic parameters is the “exterior orientation” problem.

Binocular Stereopsis

How multiple views enable one to reconstruct depth in the world

Jitendra Malik
UC Berkeley

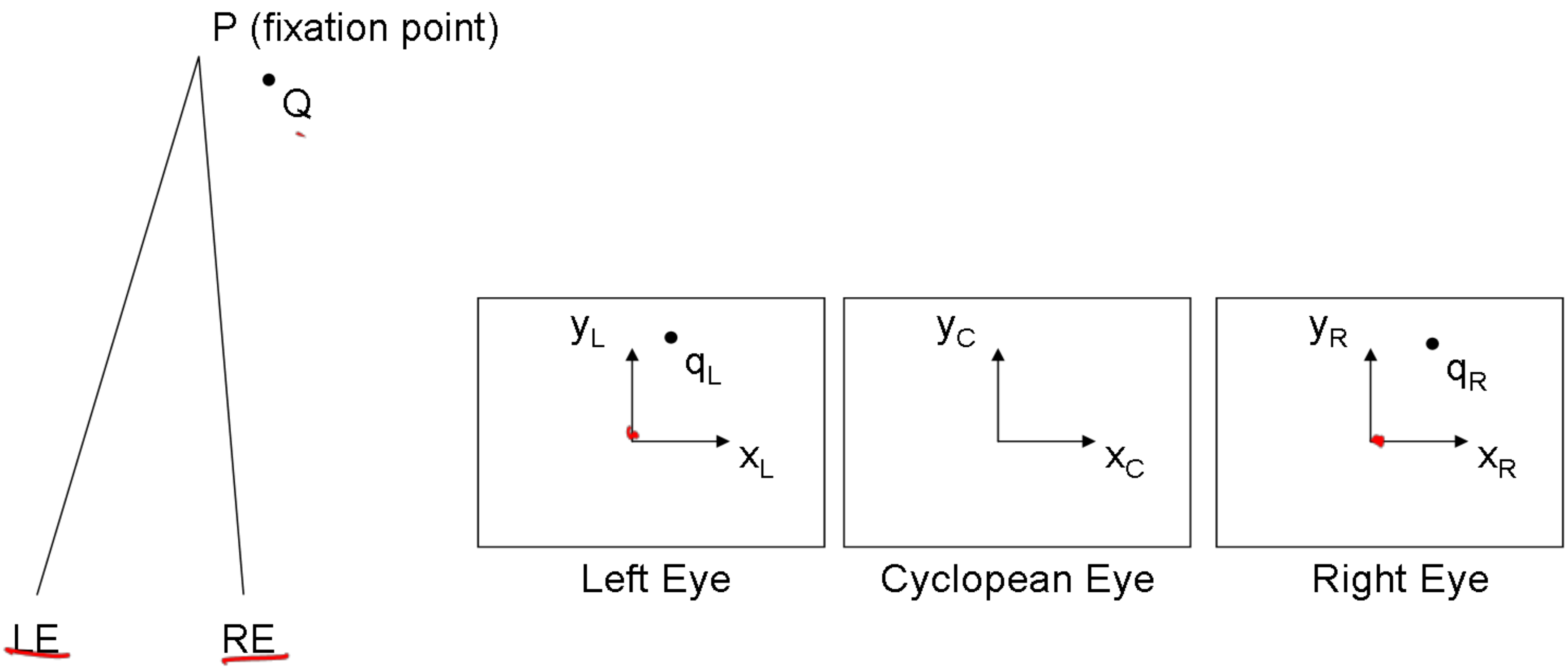
Binocular Stereopsis



Various camera configurations

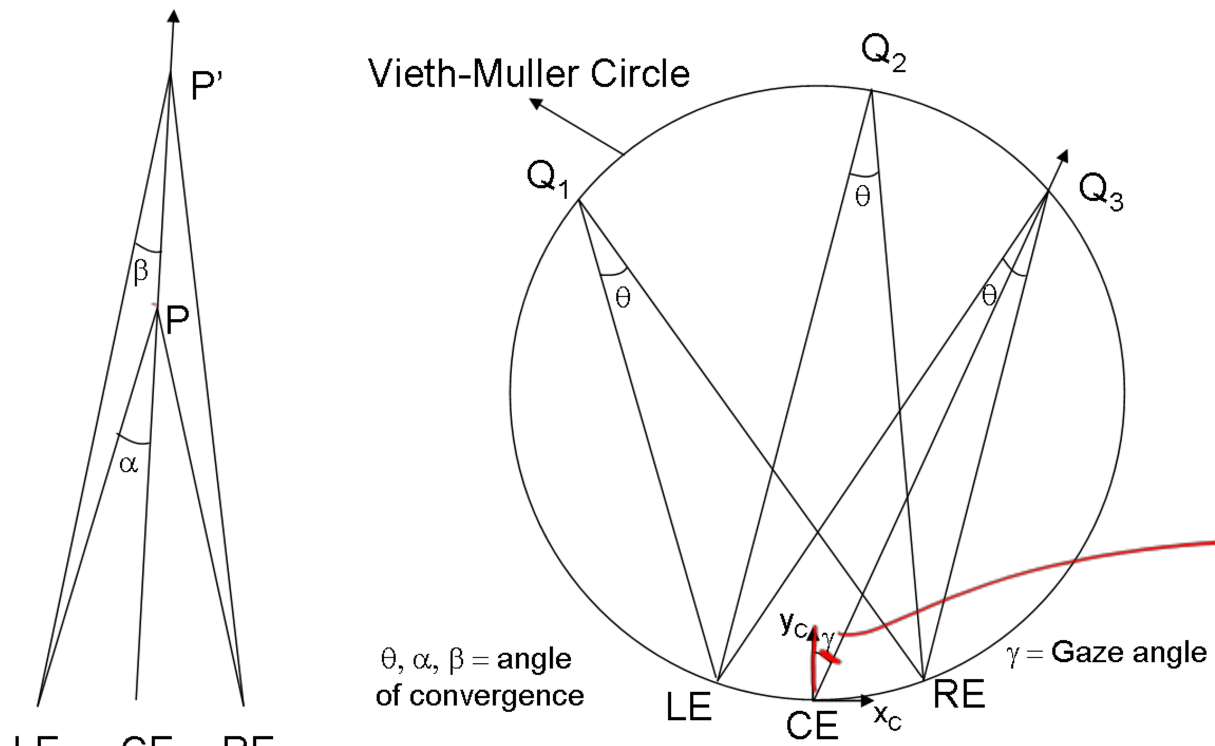
- Single point of fixation where optical axes intersect
- Optical axes parallel (fixation at infinity)
- General case

Disparity for a fixating binocular system



The Cyclopean eye is an imaginary eye at the midpoint of LE & RE

The two basic binocular eye movements



γ GAZE angle

Vergence Movement

Version Movement

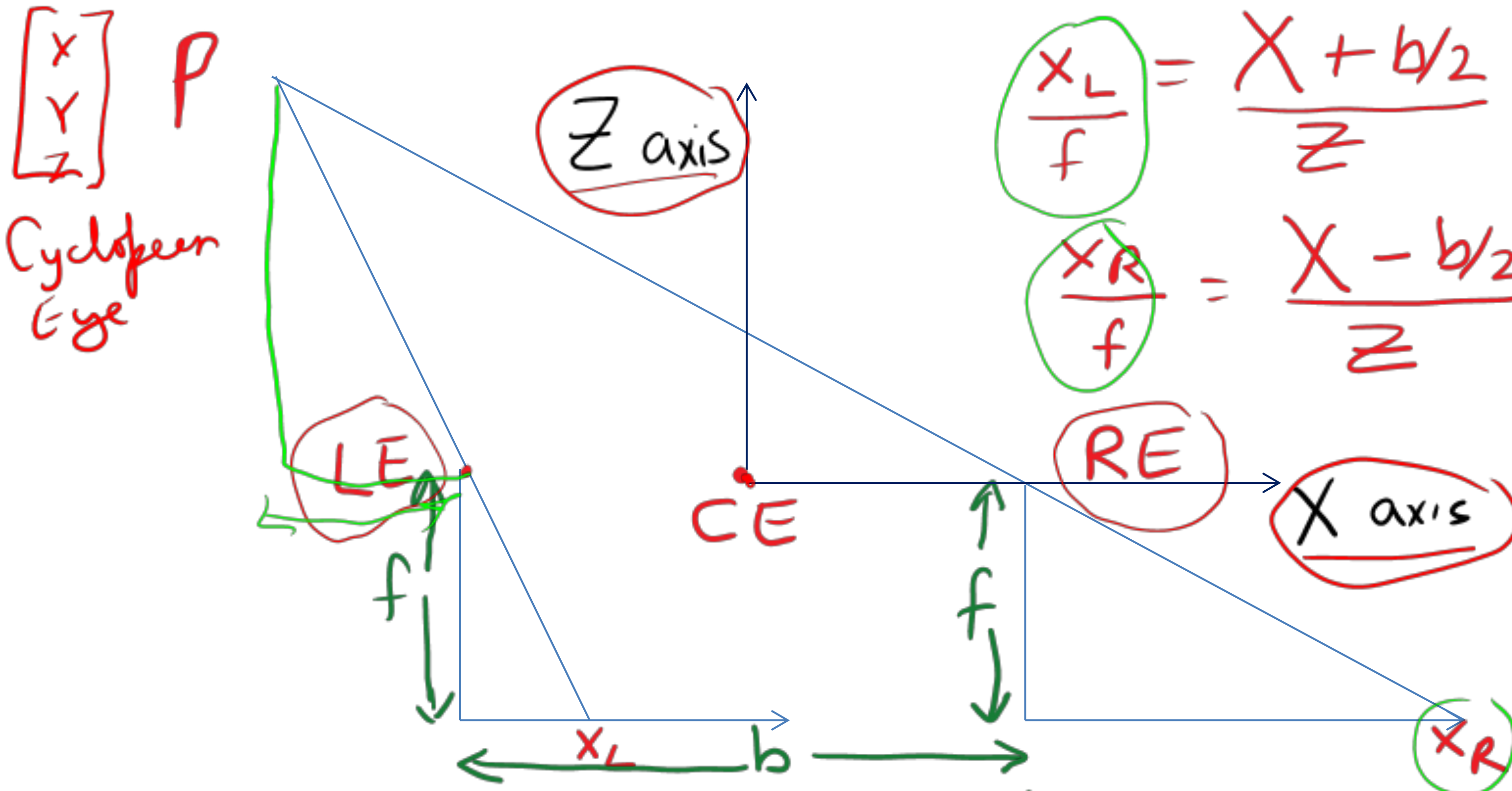
change ANGLE OF CONVERGENCE

Change GAZE

Various camera configurations

- Single point of fixation where optical axes intersect
- Optical axes parallel (fixation at infinity)
- General case

Parallel Optical Axes (fixation at infinity)

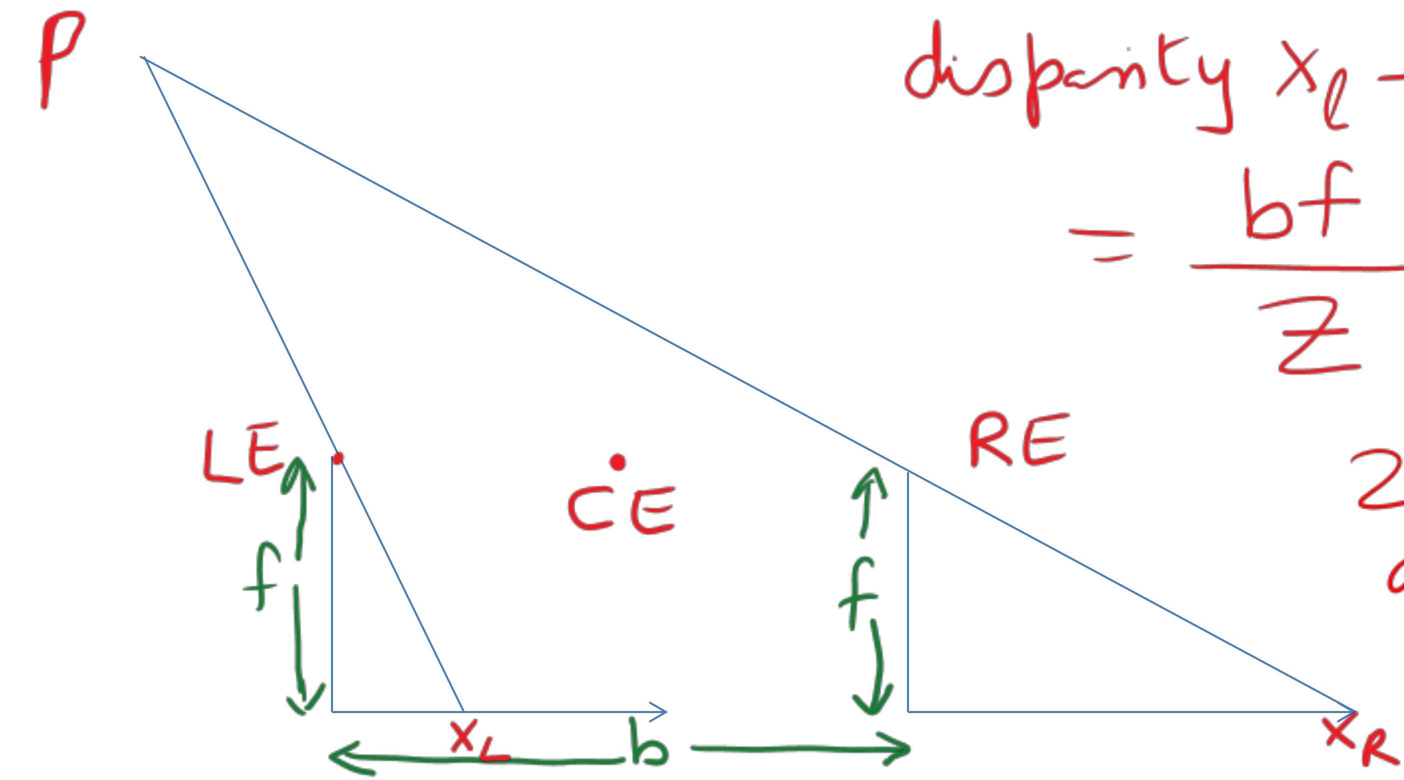


$$\frac{x_L}{f} = \frac{X + b/2}{Z}$$

$$\frac{x_R}{f} = \frac{X - b/2}{Z}$$

b baseline f focal length

Parallel Optical Axes (fixation at infinity)



disparity $x_l - x_r$

$$= \frac{bf}{Z}$$

$Z \to \infty$

disparity $\to 0$

b baseline f focal length

P in world

$P_L \leftrightarrow P_R$ CORRESPONDENCE

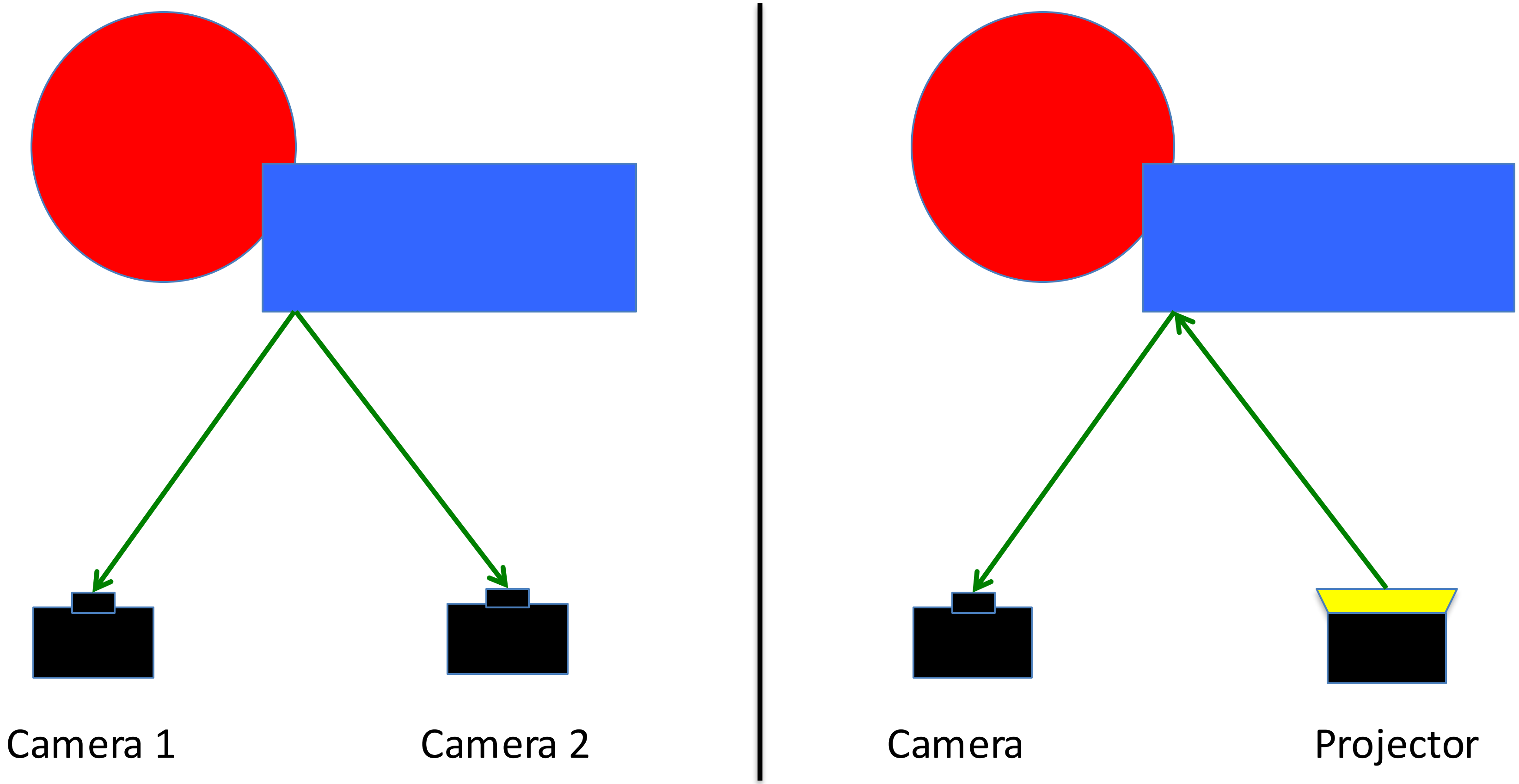
Range Sensors

primesense sensor (used in Kinect)

Velodyne LIDAR Sensor

http://www.primesense.com/, http://www.ifixit.com/,
http://mirror.umd.edu/roswiki/kinect_calibration(2f)technical.html
http://velodynelidar.com/lidar/lidar.aspx

Depth from Triangulation

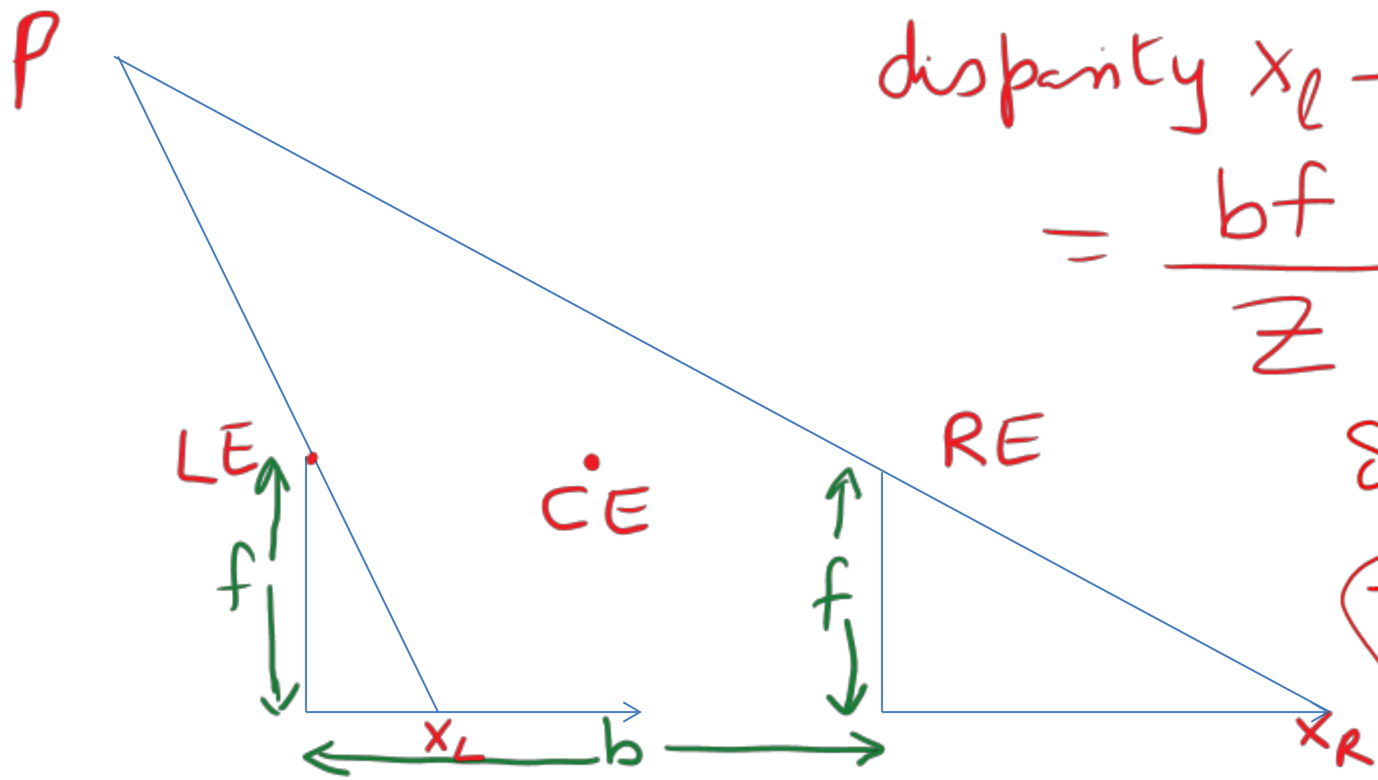


Passive Stereopsis

Active Stereopsis

Active sensing simplifies the problem of estimating point correspondences

Recall the formula for disparity with parallel optical axes…



disparity $x_l - x_r$

$$= \frac{bf}{Z}$$

$$\delta = \frac{c_1}{Z}$$

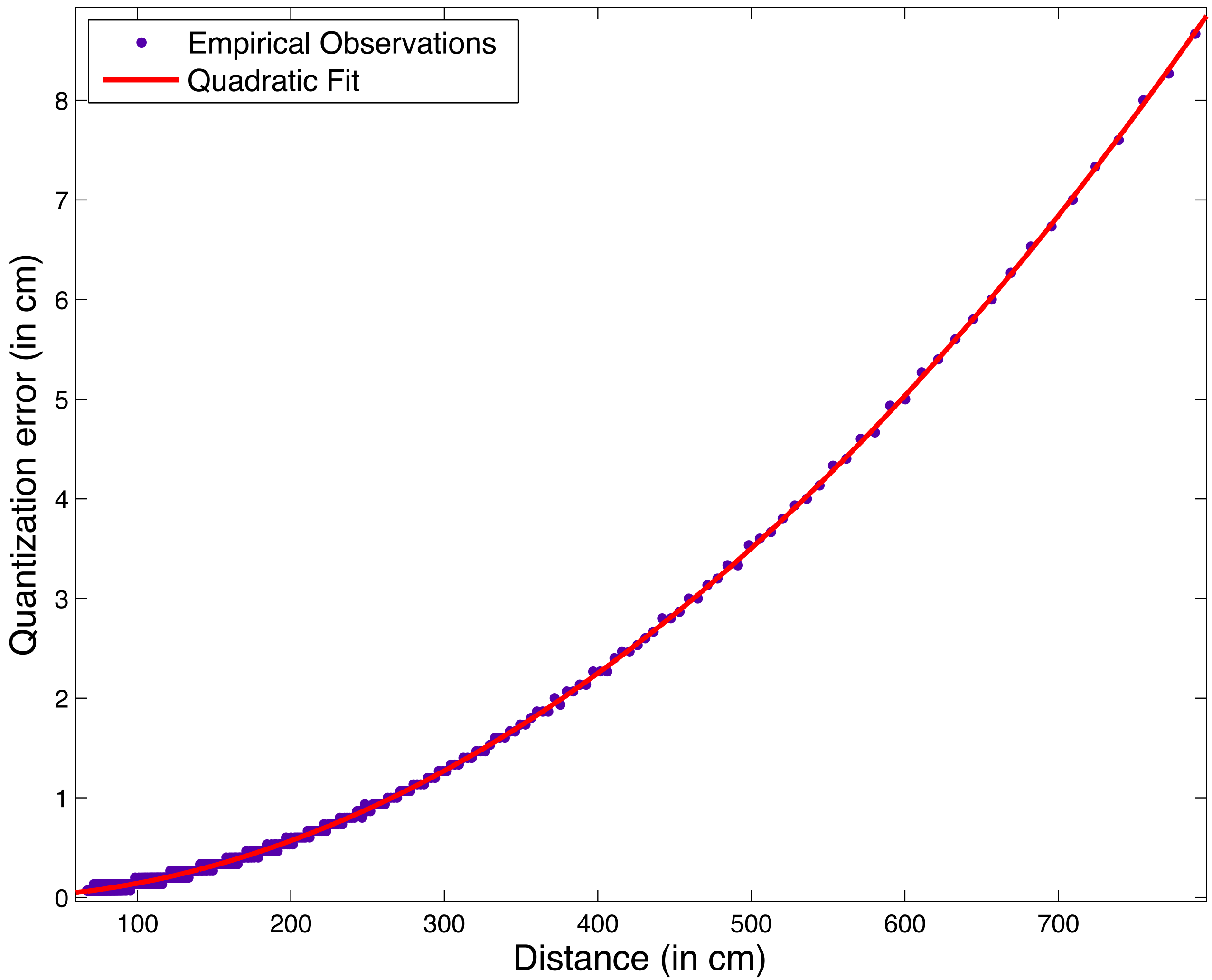
$$Z = \frac{c_2}{\delta}$$

b baseline f focal length

error(distance) – Kinect type sensor

Error in distance estimate increases quadratically with the distance

$$
\begin{aligned}
Z &= \text{distance} \\
d &= \text{disparity} \\
Z &= \frac{C}{d} \\
\delta Z &= \frac{-Z^2}{C}\delta d \\
|\delta Z| &= \frac{Z^2}{C}|\delta d| \\
\text{error} &\propto \text{distance}^2
\end{aligned}
$$



Bessel chose the star 61 Cygni as a likely star to be near the Sun, and therefore to have appreciable parallax. 61 Cygni is not nearly so bright as α Lyræ, but has a very great angular movement or proper-motion among the stars. Bessel used an instrument called a heliometer. Like Struve's telescope, it was mounted so that it could be driven by clock-work to point always at the same star. The object-glass of Bessel's telescope was made by the great optician Fraunhofer, with the intention of cutting it in halves. Fraunhofer died before the time came to carry out this delicate operation, but it was successfully accomplished after his death.

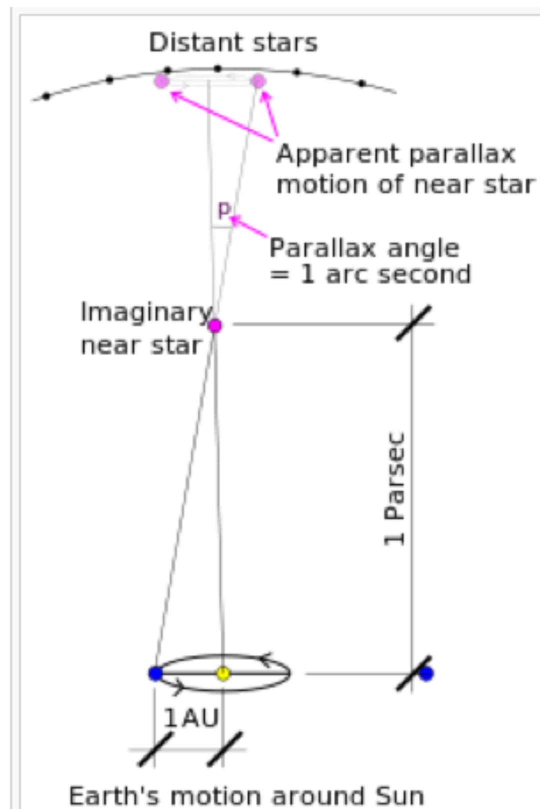
Delicate mechanism was provided for turning the glass, and also for moving the two halves relatively to each other; the amount of movement being very accurately measured by screws. Each half gives a perfect image of any object which is examined, but the two images are shifted by an amount equal to the distance one half of the lens is moved along the other. Thus when a bright star and faint star are looked at, one half of the object-glass can be made to give images S and *s*, and the other half S′ and *s*′. By moving the screw exactly the right amount *s*′ can be made to coincide with S, and the reading of the screw gives a measure of the angular distance between the two stars. Bessel made observations on 98 nights extending from August 1837 to September 1838. The following table, taken from a report by Main (*Mem. R. A. S.* vol. xii. p. 29), shows how closely the mean of the observations for each month accords with the supposition that the star has the parallax 0″·369:—

Mean date.	Observed Displacement.	Effect of parallax 0″·369.	Mean date.	Observed Displacement.	Effect of parallax 0″·369.
1837. Aug. 23	+0″·197	+0″·212	1838. Feb. 5.........	−0″·223	−0″·266
Sept. 14	+0·100	+0·100	May 14	+0·245	+0·238
Oct. 12	+0·040	−0·057	June 19......	+0·360	+0·332
Nov. 22	−0·214	−0·258	July 13	+0·216	+0·332
Dec. 21	−0·322	−0·317	Aug. 19	+0·151	+0·227
1838. Jan. 14	−0·376	−0·318	Sept. 19......	+0·040	+0·073

The great and difficult problem which had occupied astronomers for many generations was thus solved for three separate stars in 1838:—

	Parallax.	Distance.	Modern observations.	
			Parallax.	Distance.
α Centauri (Henderson)......	1″·0	200,000	0″·750	270,000
61 Cygni (Bessel)	0·314	640,000	·285	700,000
α Lyræ (Struve)	0·262	760,000	·10	2,000,000

(The unit of distance is that from the Earth to the Sun.)



Various camera configurations

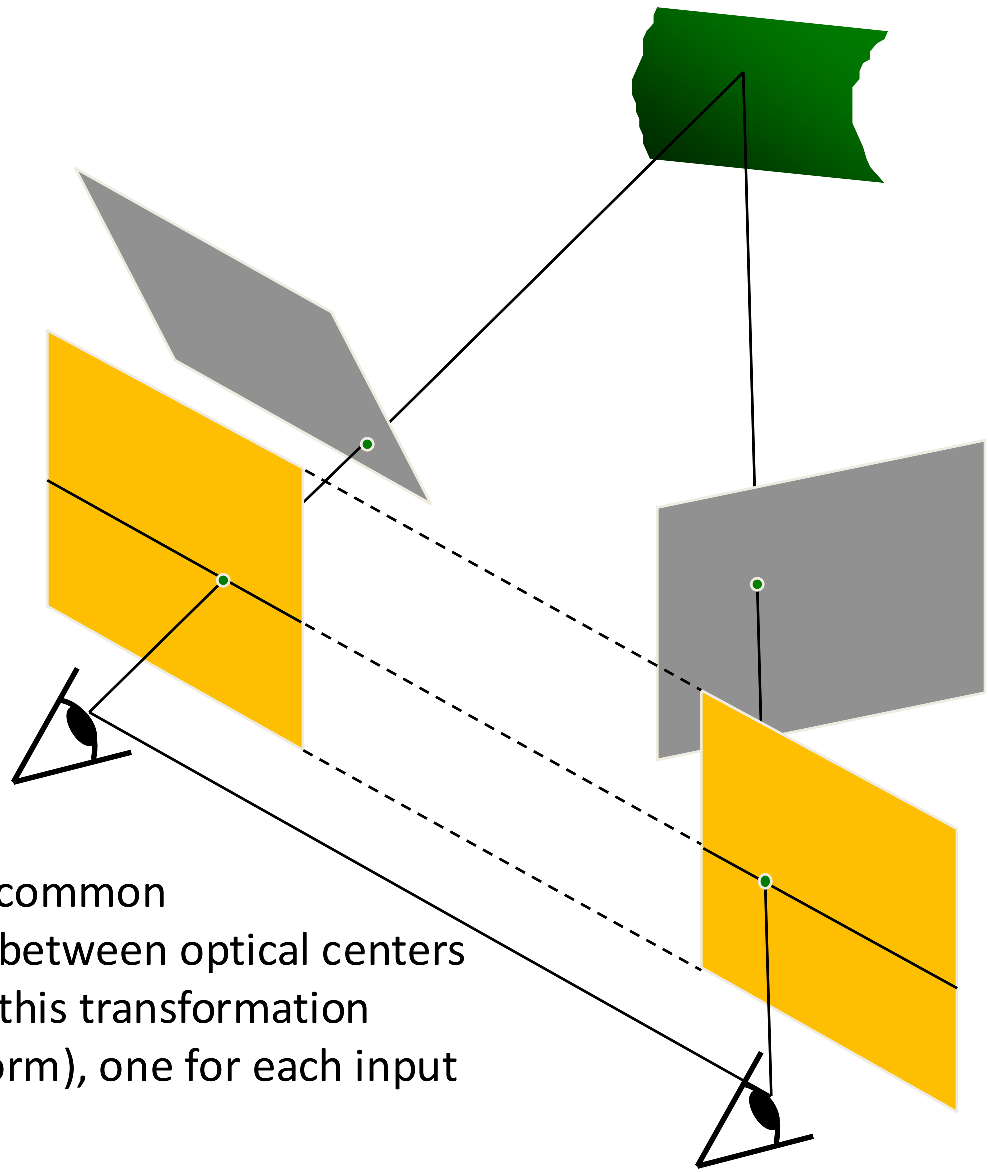
- Single point of fixation where optical axes intersect
- Optical axes parallel (fixation at infinity)
- General case

images from internet

Recover
R, t
Z

C_1 C_2
R, t

Stereo image rectification



- Reproject image planes onto a common
- plane parallel to the line between optical centers
- Pixel motion is horizontal after this transformation
- Two homographies (3x3 transform), one for each input image reprojection

•C. Loop and Z. Zhang. [Computing Rectifying Homographies for Stereo Vision](). IEEE Conf. Computer Vision and Pattern Recognition, 1999.

Rectification example