University of Zurich[UZH] | ETH zürich
Institute of Informatics – Institute of Neuroinformatics

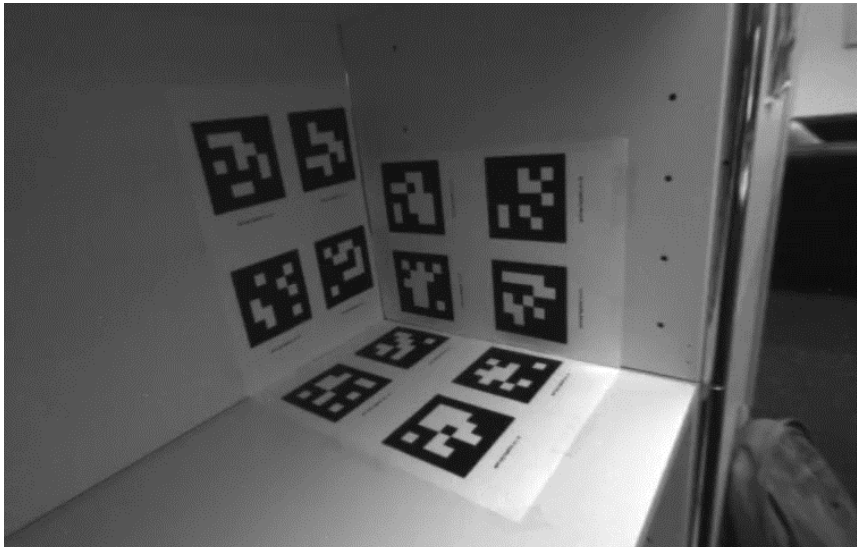
ROBOTICS & PERCEPTION GROUP

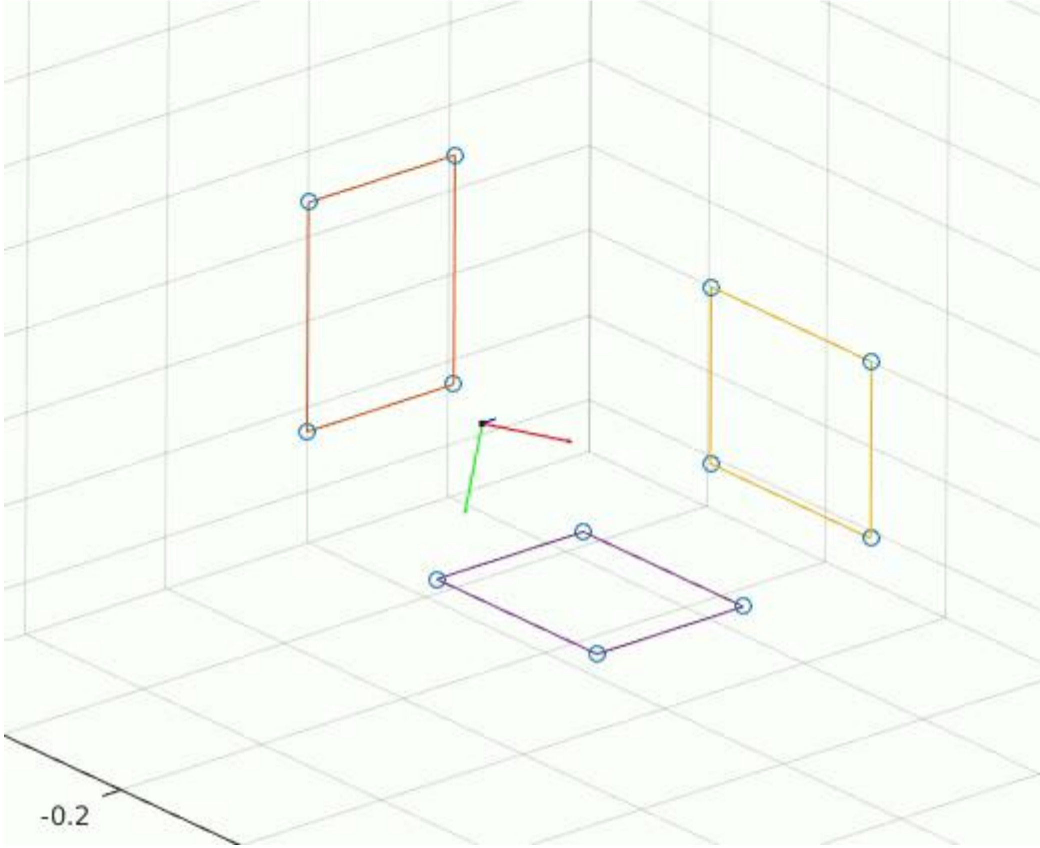
Lecture 03
Image Formation 2

Davide Scaramuzza
http://rpg.ifi.uzh.ch

Lab Exercise 2 - Today afternoon

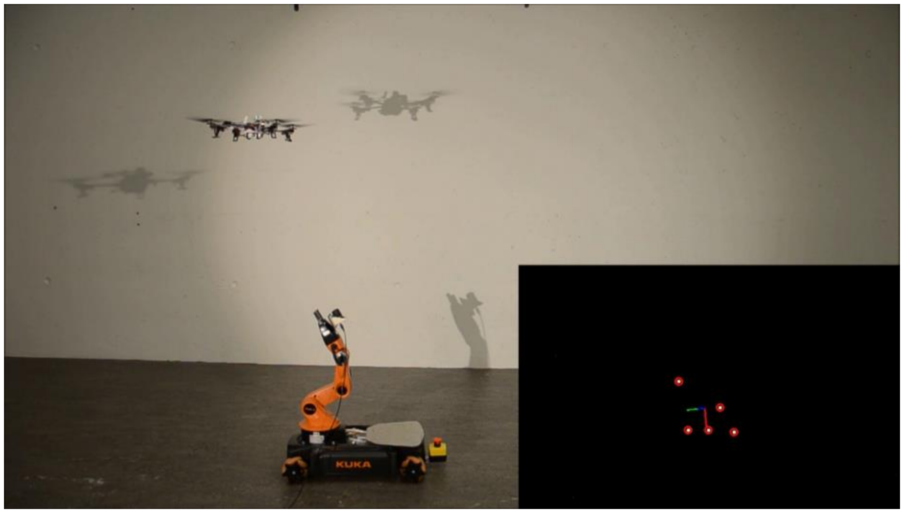
- Room ETH HG E 1.1 from 13:15 to 15:00
- Work description: your first camera motion estimator using DLT

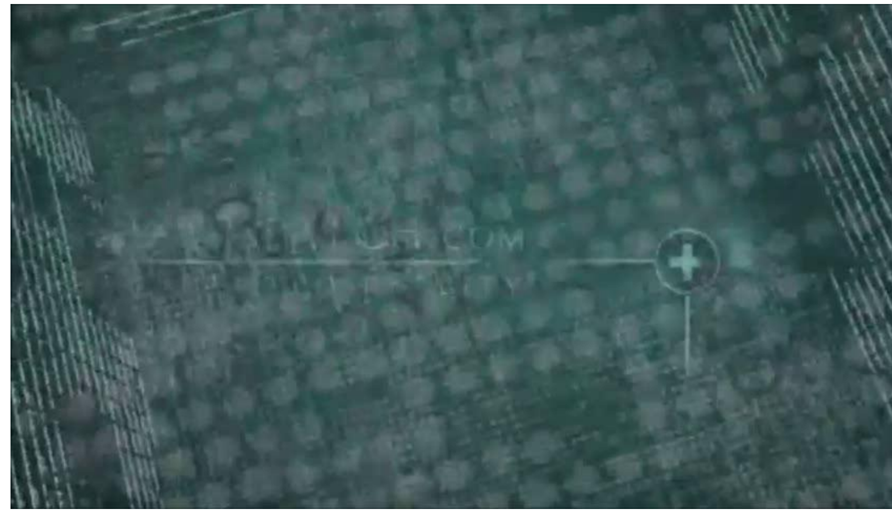


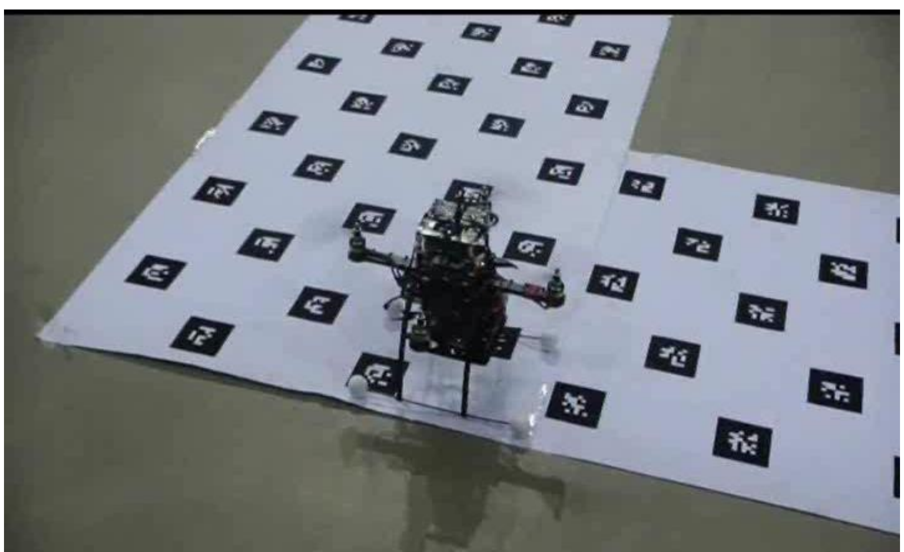


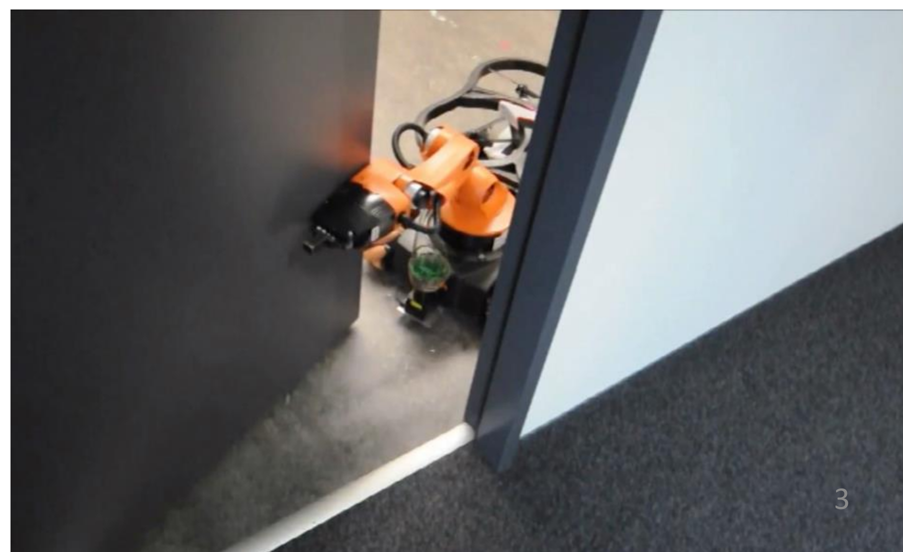
Goal of today's lecture

- Study the algorithms behind robot-position control and augmented reality







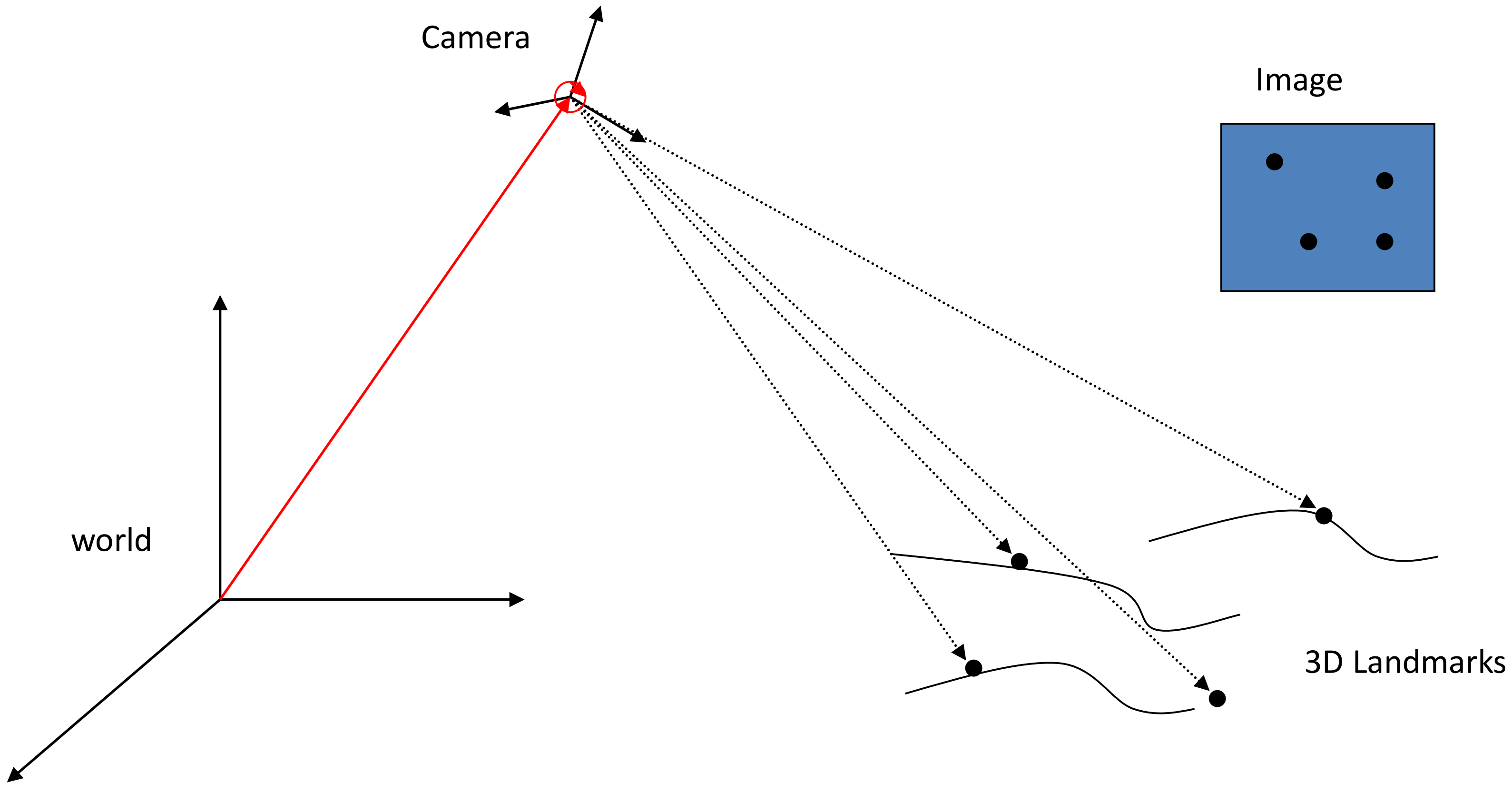


Outline of this lecture

- (Geometric) Camera calibration
  - PnP problem
    - P3P for **calibrated** cameras
    - DLT for **uncalibrated** cameras
- Omnidirectional cameras

Perspective from *n* Points (aka PnP Problem)

- Given known 3D landmarks positions in the world frame and given their image correspondences in the camera frame, determine the 6DOF pose of the camera in the world frame (including the intrinsinc parameters if uncalibrated)



Perspective from *n* Points (aka PnP Problem)

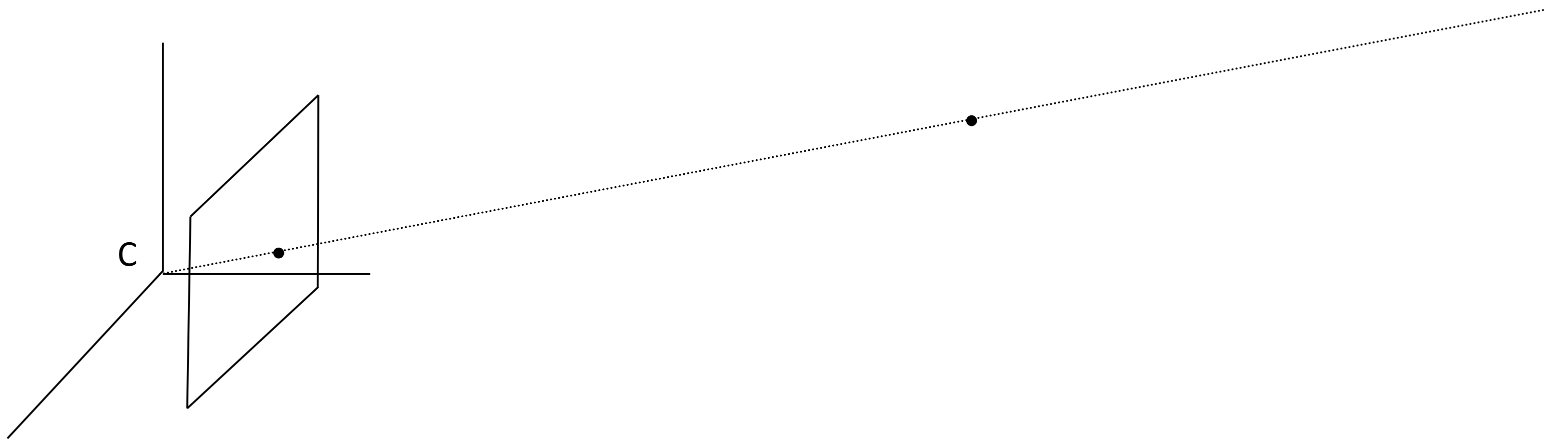
PnP Problem

Calibrated camera **(i.e., instrinc parameters are known)**	**Uncalibrated camera** **(i.e., intrinsic parameters unknown)**
Works for any 3D point configurations	Direct Linear Transform (DLT)
Minimum number of points: 3 P3P (Perspective from Three Points)	Minimum number of points: 4 if coplanar 6 if non coplanar

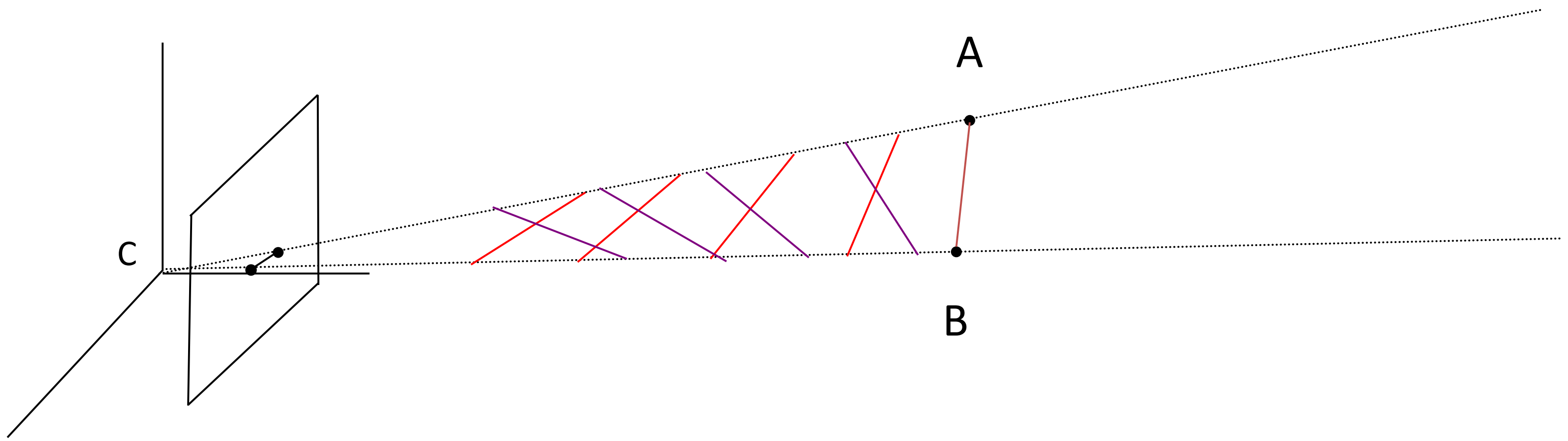
How Many Points are Enough?

- 1 Point: infinitely many solutions.
- 2 Points: infinitely many solutions, but bounded.
- 3 Points:
  - (no 3 collinear) finitely many solutions (up to 4).
- 4 Points:
  - Unique solution

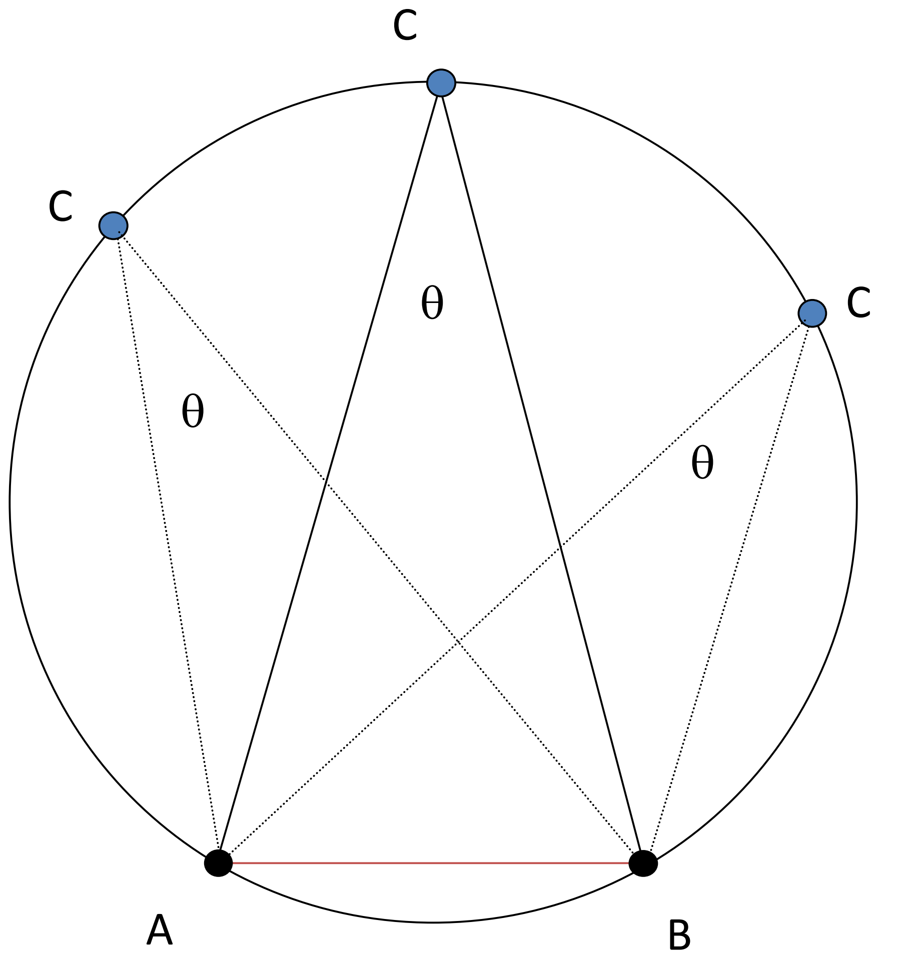
1 Point



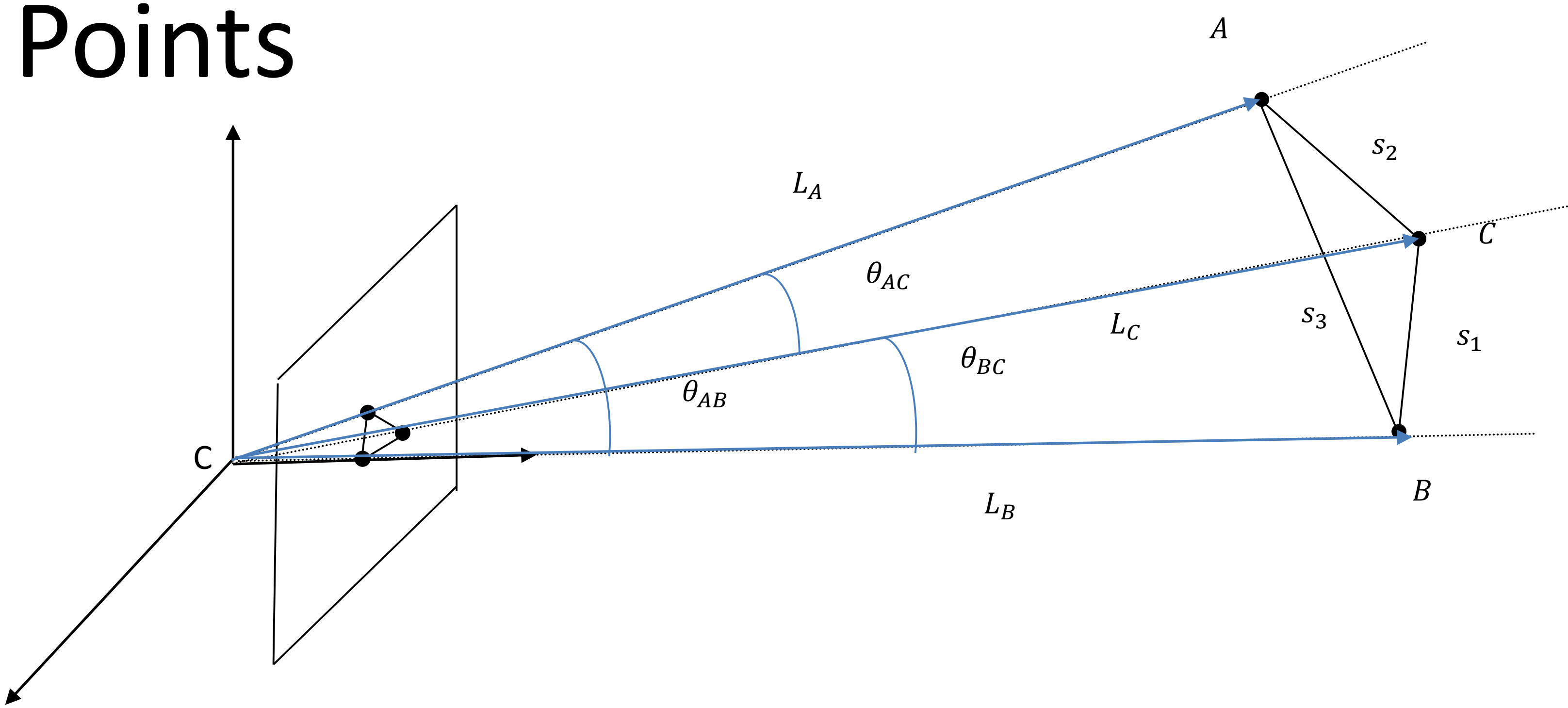
2 Points



Inscribed Angles are Equal



3 Points



From Carnot's Theorem:

$$s_1^2 = L_B^2 + L_C^2 - 2L_BL_C\cos\theta_{BC}$$

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

$$s_3^2 = L_A^2 + L_B^2 - 2L_AL_B\cos\theta_{AB}$$

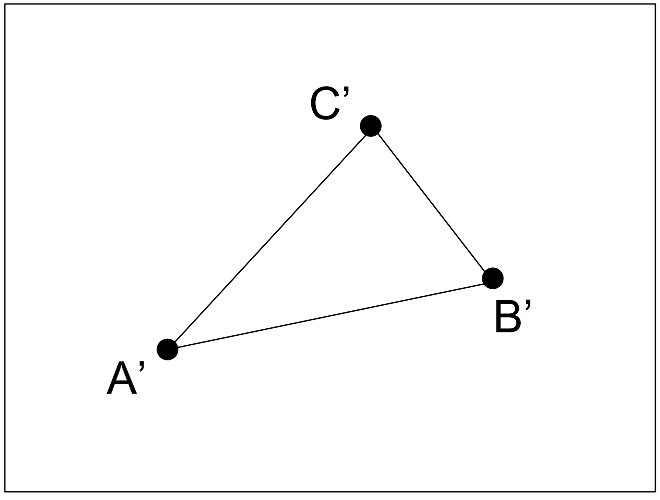


Image Plane

Algebraic Approach: reduce to 4th order equation

(Fischler and Bolles, 1981)

$$s_1^2 = L_B^2 + L_C^2 - 2L_B L_C \cos\theta_{BC}$$

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

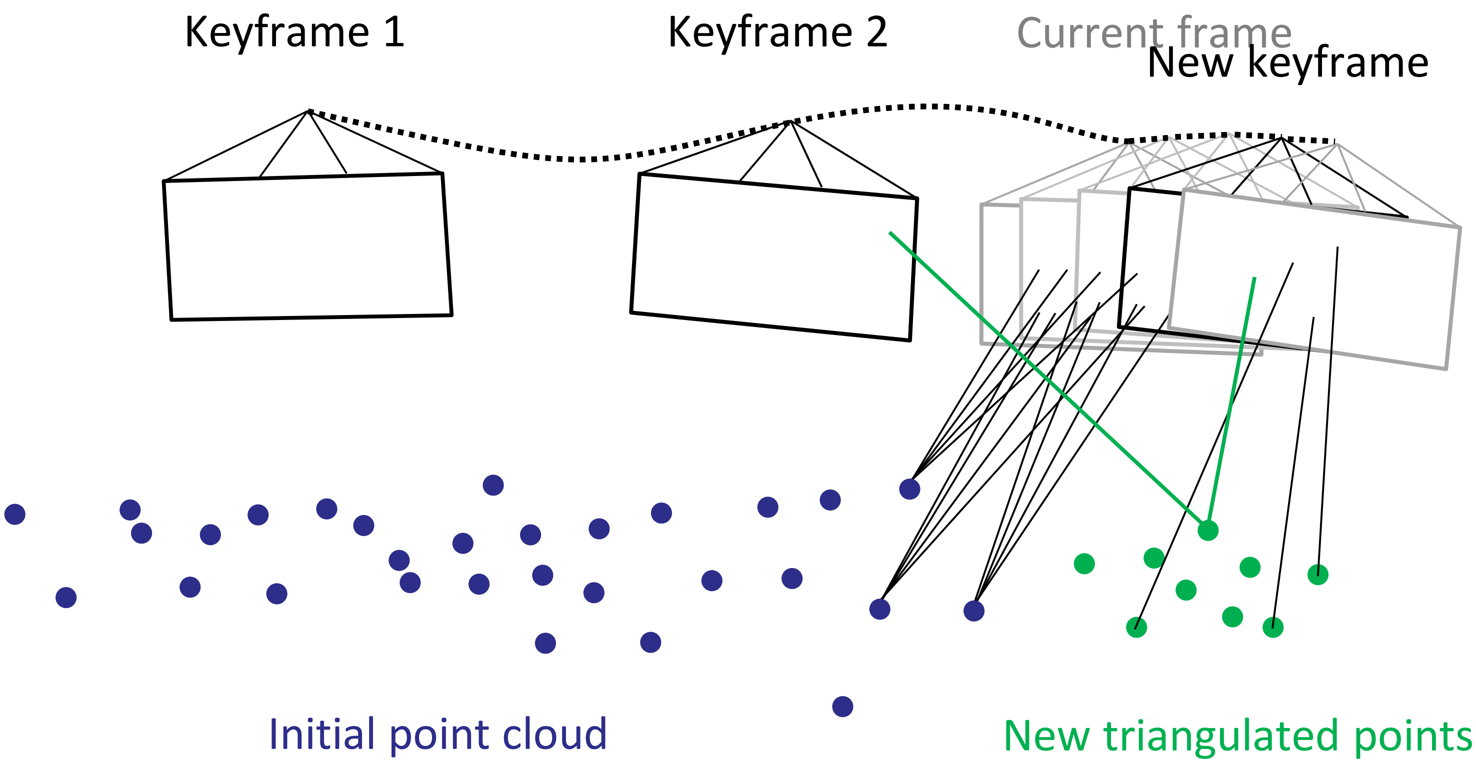
$$s_3^2 = L_A^2 + L_B^2 - 2L_A L_B \cos\theta_{AB}$$

- 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.
- A 4th point can be used to disambiguate the solutions.

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

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

Application to Monocular Visual Odometry: camera pose estimation from known 3D-2D correspondences



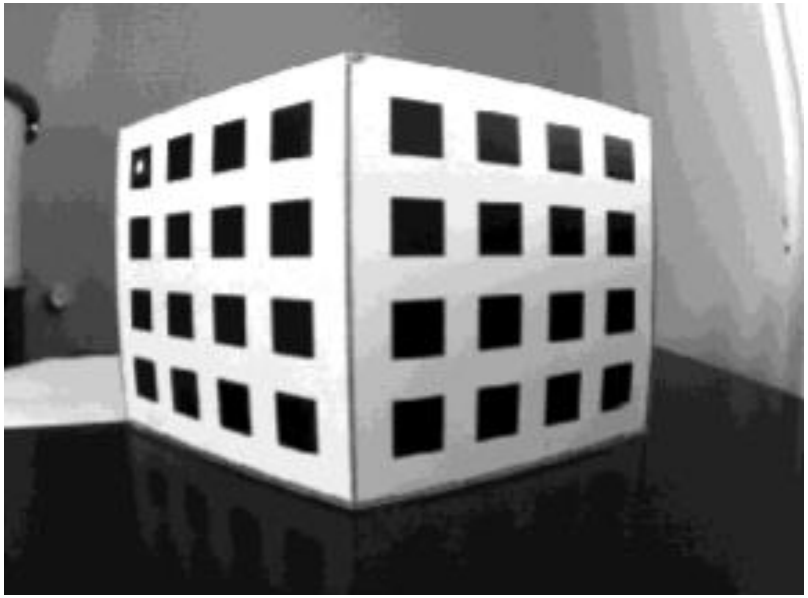
AR Application: Microsoft HoloLens

Outline of this lecture

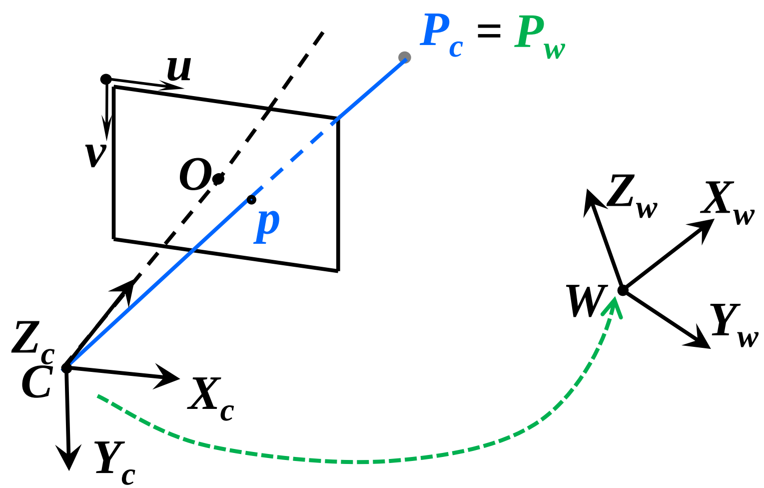
- (Geometric) Camera calibration
  - PnP problem
    - P3P for **calibrated** cameras
    - DLT for **uncalibrated** cameras
      - From general 3D objects
      - From planar grids
- Omnidirectional cameras

Camera calibration

- Calibration is the process to determine the **intrinsic and extrinsic** parameters of the camera model
- A method proposed in 1987 by Tsai consists of measuring the 3D position of $n \geq 6$ control points on a three-dimensional calibration target and the 2D coordinates of their projection in the image. This problem is also called "**Resection**", or "**Perspective from n Points (PnP)**", or "**Camera pose from 3D-to-2D correspondences**", and is one of the most widely used algorithms in Computer Vision and Robotics
- Solution: The intrinsic and extrinsic parameters are computed directly from the perspective projection equation; let's see how!



3D position of control points is assigned in a reference frame specified by the user



Camera calibration: Direct Linear Transform (DLT)

Our goal is to compute K, R, and T that satisfy the perspective projection equation (we neglect the radial distortion)

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

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

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

Camera calibration: Direct Linear Transform (DLT)

Our goal is to compute K, R, and T that satisfy the perspective projection equation (we neglect the radial distortion)

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

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

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

Camera calibration: Direct Linear Transform (DLT)

Our goal is to compute K, R, and T that satisfy the perspective projection equation (we neglect the radial distortion)

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

$$\Rightarrow \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = M \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} m_1^T \\ m_2^T \\ m_3^T \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

where m_i^T is the i-*th* row of M

Camera calibration: Direct Linear Transform (DLT)

$$\Rightarrow \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} m_1^T \\ m_2^T \\ m_3^T \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \rightarrow P$$

Conversion back from homogeneous coordinates to pixel coordinates leads to:

$$u = \frac{\tilde{u}}{\tilde{w}} = \frac{m_1^T \cdot P}{m_3^T \cdot P} \qquad \Rightarrow \qquad (m_1^T - u_i m_3^T) \cdot P_i = 0$$

$$v = \frac{\tilde{v}}{\tilde{w}} = \frac{m_2^T \cdot P}{m_3^T \cdot P} \qquad\qquad (m_2^T - v_i m_3^T) \cdot P_i = 0$$

Camera calibration: Direct Linear Transform (DLT)

By re-arranging the terms, we obtain

$$\begin{matrix}(m_1^T - u_i m_3^T)\cdot P_i = 0 \\ (m_2^T - v_i m_3^T)\cdot P_i = 0\end{matrix} \Rightarrow \begin{pmatrix} P_1^T & 0^T & -u_1 P_1^T \\ 0^T & P_1^T & -v_1 P_1^T \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ m_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

For n points, we can stack all these equations into a big matrix:

$$\begin{pmatrix} P_1^T & 0^T & -u_1 P_1^T \\ 0^T & P_1^T & -v_1 P_1^T \\ \cdots & \cdots & \cdots \\ P_n^T & 0^T & -u_n P_n^T \\ 0^T & P_n^T & -v_n P_n^T \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ m_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}$$

Camera calibration: Direct Linear Transform (DLT)

By re-arranging the terms, we obtain

$$\begin{aligned}(m_1^T - u_i m_3^T)\cdot P_i = 0\\ (m_2^T - v_i m_3^T)\cdot P_i = 0\end{aligned} \Rightarrow \begin{pmatrix} P_1^T & 0^T & -u_1 P_1^T \\ 0^T & P_1^T & -v_1 P_1^T \end{pmatrix}\begin{pmatrix} m_1 \\ m_2 \\ m_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

For n points, we can stack all these equations into a big matrix:

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

Q (this matrix is **known**)

M (this matrix is **unknown**)

Camera calibration: Direct Linear Transform (DLT)

$$Q \cdot M = 0$$

Minimal solution

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

Over-determined solution

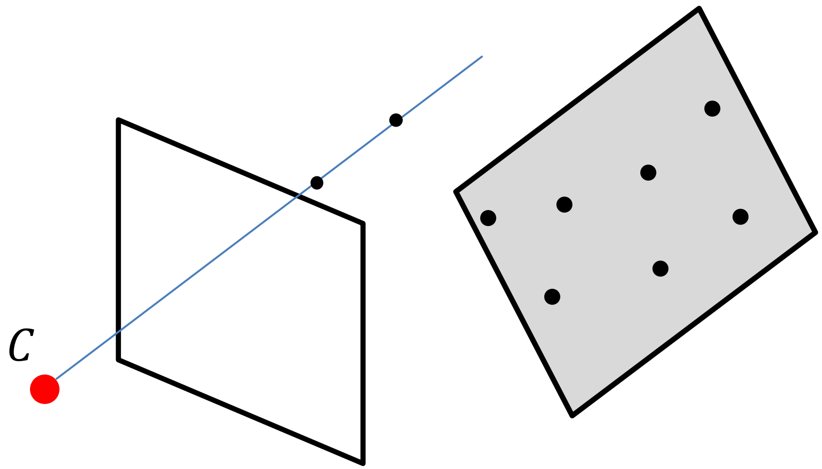
- *n* ≥ 6 points
- A solution is to minimize $||QM||^2$ subject to the constraint $||M||^2 = 1$.
  It can be solved through Singular Value Decomposition (SVD). The solution is the eigenvector corresponding to the smallest eigenvalue of the matrix Q^TQ (because it is the unit vector x that minimizes $||Qx||^2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

Camera calibration: Direct Linear Transform (DLT)

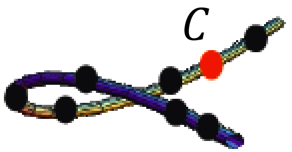
$$Q \cdot M = 0$$

Degenerate configurations

1. Points lying on a **plane** and/or along a single **line** passing through the **center of projection**



2. Camera and points on a twisted cubic (i.e., smooth curve in 3D space of degree 3)



Camera calibration: Direct Linear Transform (DLT)

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

$$M = K(R \mid T)$$

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

Camera calibration: Direct Linear Transform (DLT)

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

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

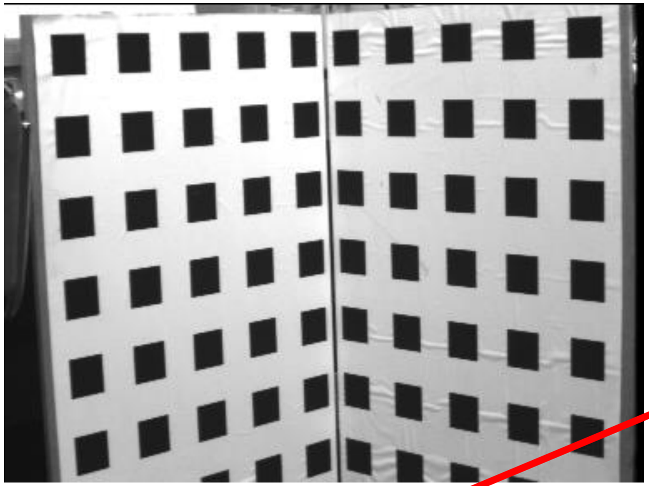
$$\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} = \begin{bmatrix} \alpha r_{11} + u_0 r_{31} & \alpha r_{12} + u_0 r_{32} & \alpha r_{13} + u_0 r_{33} & \alpha t_1 + u_0 t_3 \\ \alpha r_{21} + v_0 r_{31} & \alpha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

- However, notice that we are not enforcing the constraint that R is orthogonal, i.e., $R \cdot R^T = I$

- To do this, we can use the so-called QR factorization of M, which decomposes M into a R (orthogonal), T, and an upper triangular matrix (i.e., K)

Tsai's (1987) Calibration example

1. Edge detection
2. Straight line fitting to the detected edges
3. Intersecting the lines to obtain the image corners (corner accuracy < 0.1 pixels)
4. **Use more than 6 points** (ideally more than 20) **and not all lying on the same plane**



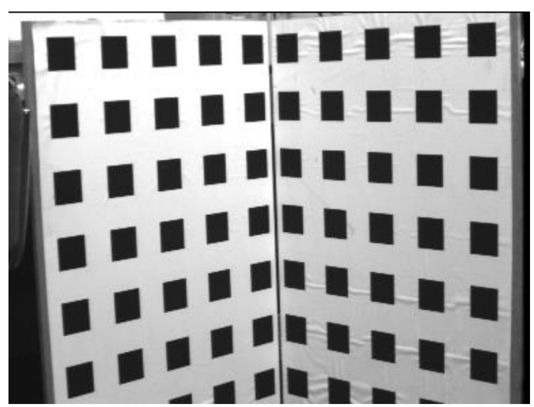
Why is this ratio not 1?

What are the «skew» and «residuals»?

α_u	α_u/α_v	skew	u_0	v_0	residual
1673.3	1.0063	1.39	379.96	305.78	0.365

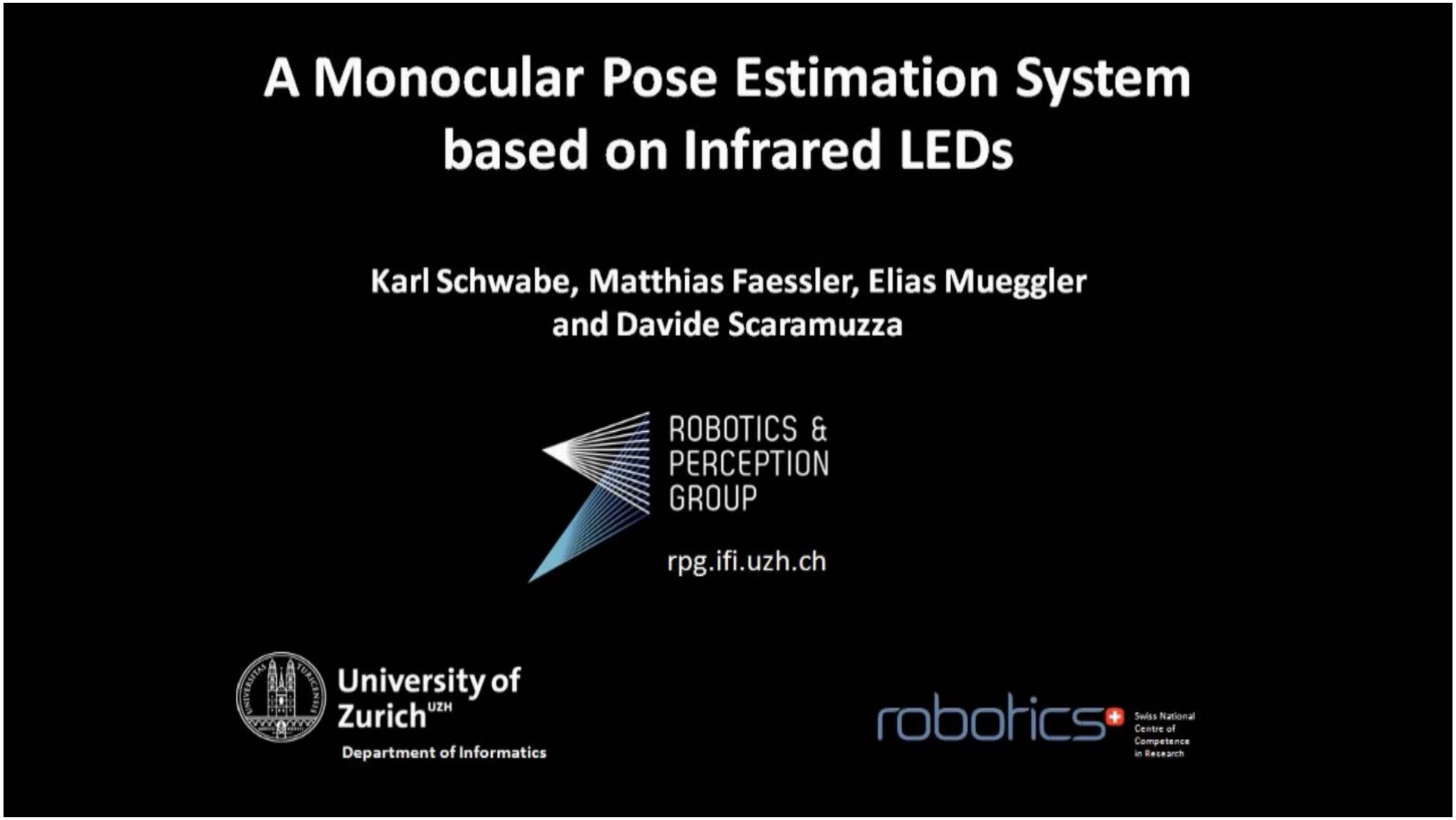
Tsai's (1987) Calibration example

- The original Tsai calibration (1987) used to consider two different focal lengths α_u, α_v (which means that the pixels are not squared) and a skew factor ($K_{12} \neq 0$, which means the pixels are parallelograms instead of rectangles) to account for possible misalignments (small x, y rotation) between image plane and lens
- Most today's cameras are well manufactured, thus, we can assume $\frac{\alpha_u}{\alpha_v} = 1$ and $K_{12} = 0$
- What is the residual? The residual is the *average* "reprojection error" (see Lecture 8). The reprojection error is computed as the distance (in pixels) between the observed pixel point and the camera-reprojected 3D point. The reprojection error gives as a quantitative measure of the accuracy of the calibration (ideally it should be zero).



f_y	f_x/f_y	skew	x_0	y_0	residual
1673.3	1.0063	1.39	379.96	305.78	0.365

DLT algorithm applied to mutual robot localization



In this case, the camera has been pre-calibrated (i.e., K is known). Can you think of how the DLT algorithm could be modified so that only R and T need to be determined and not K?

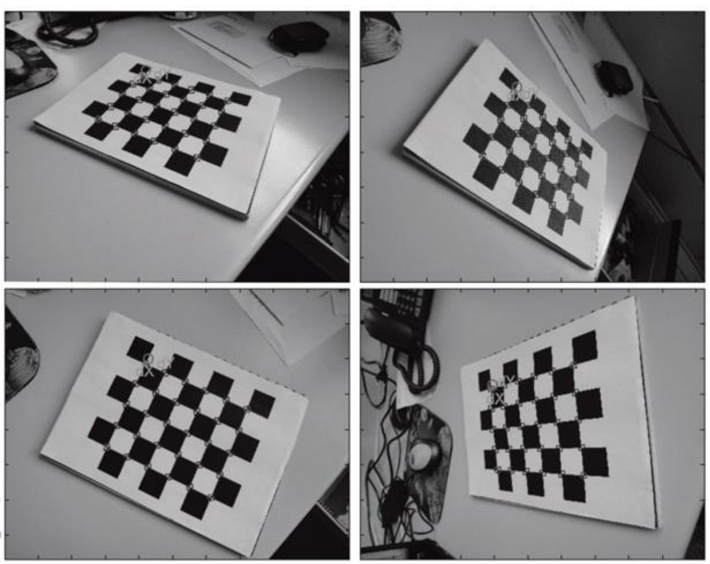
http://youtu.be/8Ui3MoOxcPQ

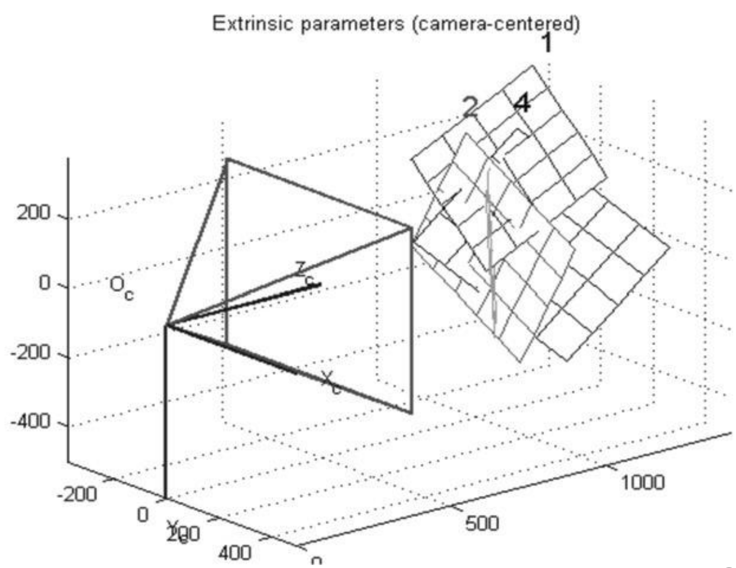
Outline of this lecture

- (Geometric) Camera calibration
  - PnP problem
    - P3P for **calibrated** cameras
    - DLT for **uncalibrated** cameras
      - From general 3D objects
      - From planar grids
- Omnidirectional cameras

Camera calibration from planar grids: homographies

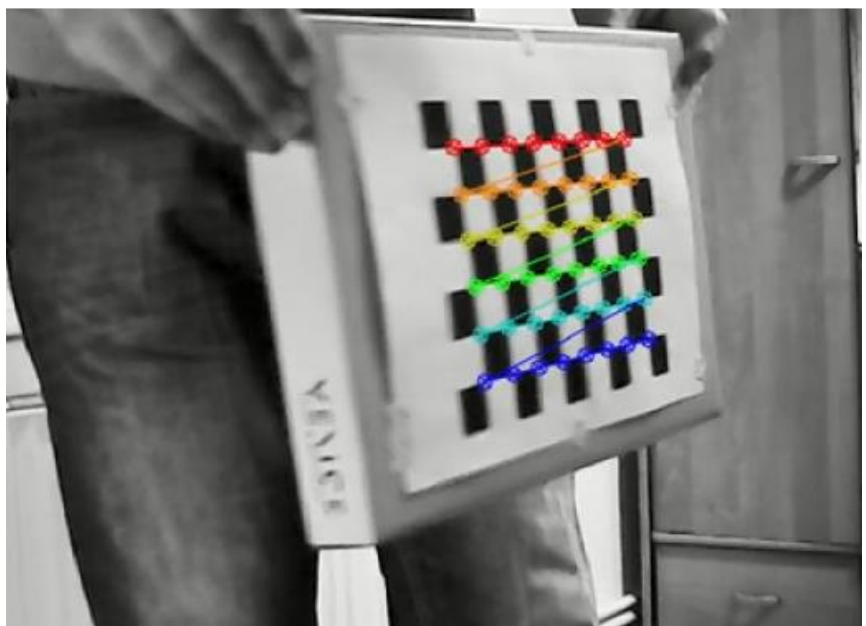
- Tsai's calibration is based on DLT algorithm, which requires points not to lie on the same plane
- An alternative method (today's standard camera calibration method) consists of using a planar grid (e.g., a chessboard) and a few images of it shown at different orientations
- This method was invented by Zhang (1999) @Microsoft Research and is implemented in both Matlab and the OpenCV C/C++ library





Camera calibration from planar grids: homographies

- Tsai's calibration is based on DLT algorithm, which requires points not to lie on the same plane
- An alternative method (today's standard camera calibration method) consists of using a planar grid (e.g., a chessboard) and a few images of it shown at different orientations
- This method was invented by Zhang (1999) @Microsoft Research and is implemented in both Matlab and the OpenCV C/C++ library



Camera calibration from planar grids: homographies

- Our goal is to compute K, R, and T, that satisfy the perspective projection equation (we neglect the radial distortion)
- Since the points lie on a plane, we have $Z_w = 0$

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

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

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

Camera calibration from planar grids: homographies

- Our goal is to compute K, R, and T, that satisfy the perspective projection equation (we neglect the radial distortion)
- Since the points lie on a plane, we have $Z_w = 0$

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

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

This matrix is called Homography

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

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

Camera calibration from planar grids: homographies

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

Conversion back from homogeneous coordinates to pixel coordinates leads to:

$$u = \frac{\tilde{u}}{\tilde{w}} = \frac{h_1^T \cdot P}{h_3^T \cdot P} \qquad v = \frac{\tilde{v}}{\tilde{w}} = \frac{h_2^T \cdot P}{h_3^T \cdot P} \quad \Rightarrow \quad \begin{aligned} (h_1^T - u_i h_3^T) \cdot P_i &= 0 \\ (h_2^T - v_i h_3^T) \cdot P_i &= 0 \end{aligned}$$

where P = $(X_w, Y_w, 1)^T$

Camera calibration from planar grids: homographies

By re-arranging the terms, we obtain

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

For n points, we can stack all these equations into a big matrix:

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

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

Camera calibration from planar grids: homographies

$$Q \cdot H = 0$$

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)

Solving for K, R and T

- H can be decomposed by recalling that

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

How to recover *K*, *R*, *T* from *H*?

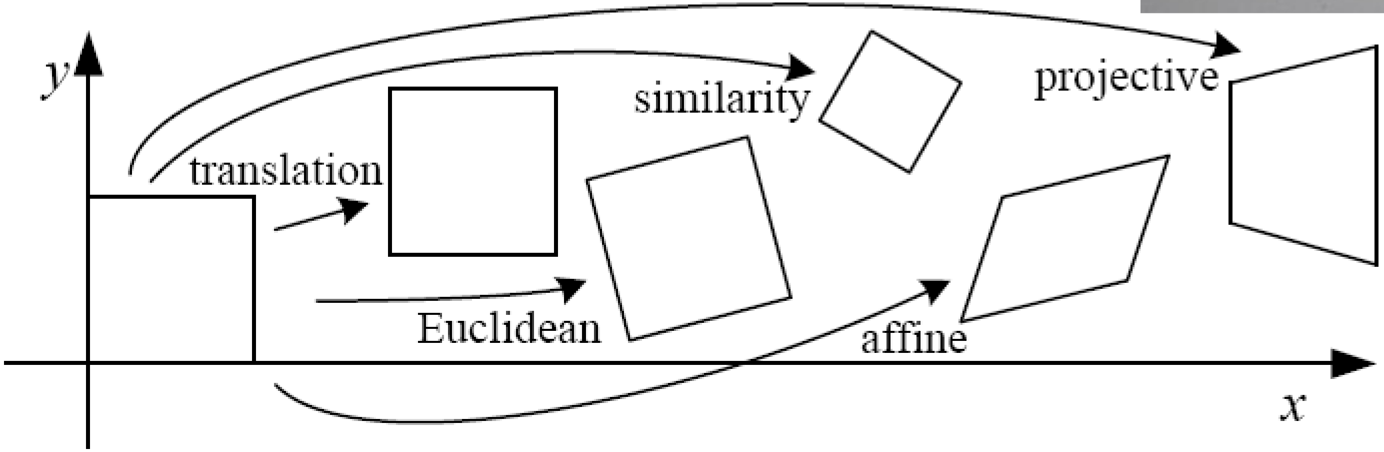
Won't be asked at the exam

1. Estimate the homography H_i for each view, using the DLT algorithm.

2. Determine the intrinsics K of the camera from a set of homographies:
   1. Each homography $H_i \sim K(\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{t})$ provides two *linear* equations in the 6 entries of the matrix $\mathrm{B} \coloneqq K^{-\top}K^{-1}$. Letting $\boldsymbol{w}_1 \coloneqq K\boldsymbol{r}_1$, $\boldsymbol{w}_2 \coloneqq K\boldsymbol{r}_2$, the rotation constraints $\boldsymbol{r}_1^\top \boldsymbol{r}_1 = \boldsymbol{r}_2^\top \boldsymbol{r}_2 = 1$ and $\boldsymbol{r}_1^\top \boldsymbol{r}_2 = 0$ become $\boldsymbol{w}_1^\top B \boldsymbol{w}_1 - \boldsymbol{w}_2^\top B \boldsymbol{w}_2 = 0$ and $\boldsymbol{w}_1^\top B \boldsymbol{w}_2 = 0$.
   2. Stack 2*N* equations from *N* views, to yield a linear system $A\boldsymbol{b} = \boldsymbol{0}$. Solve for ***b*** (i.e., *B*) using the Singular Value Decomposition (SVD).
   3. Use Cholesky decomposition to obtain *K* from *B*.

3. The extrinsic parameters for each view can be computed using *K*: $\boldsymbol{r}_1 \sim \lambda K^{-1} H_i(:,1)$, $\boldsymbol{r}_2 \sim \lambda K^{-1} H_i(:,2)$, $\boldsymbol{r}_3 = \boldsymbol{r}_1 \times \boldsymbol{r}_2$ and $T_i = \lambda K^{-1} H_i(:,3)$, with $\lambda = 1/K^{-1} H_i(:,1)$. Finally, build $R_i = (\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3)$ and enforce rotation matrix constraints.

Types of 2D Transformations

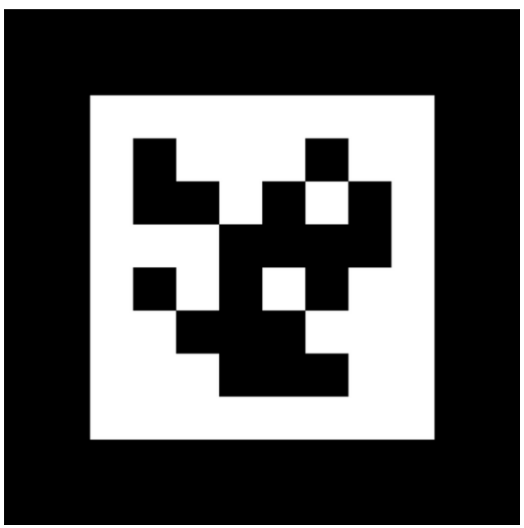


Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} I \mid t \end{bmatrix}_{2\times 3}$	2	orientation + ···	
rigid (Euclidean)	$\begin{bmatrix} R \mid t \end{bmatrix}_{2\times 3}$	3	lengths + ···	
similarity	$\begin{bmatrix} sR \mid t \end{bmatrix}_{2\times 3}$	4	angles + ···	
affine	$\begin{bmatrix} A \end{bmatrix}_{2\times 3}$	6	parallelism + ···	
projective	$\begin{bmatrix} \tilde{H} \end{bmatrix}_{3\times 3}$	8	straight lines	

This transformation is called Homography

Application of calibration from planar grids

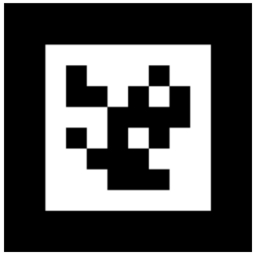
- Today, there are thousands of application of this algorithm:
  - Augmented reality

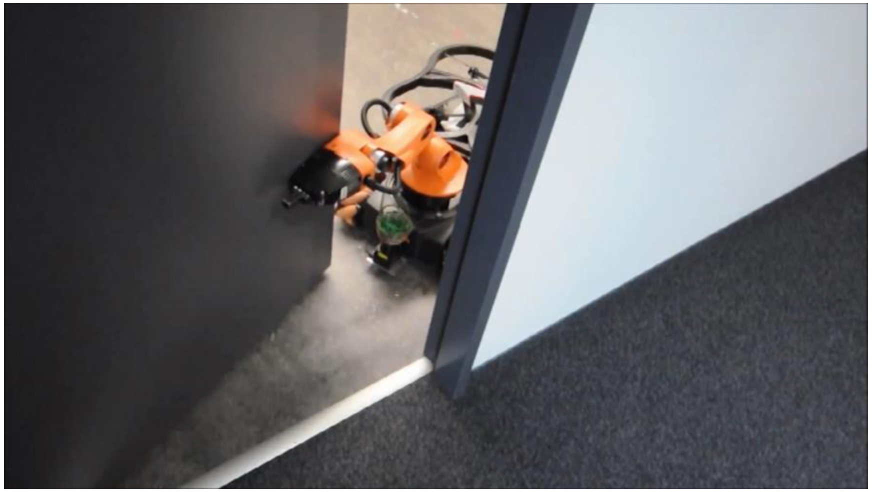


Most AR apps use [AprilTag] or [ARuco Markers]

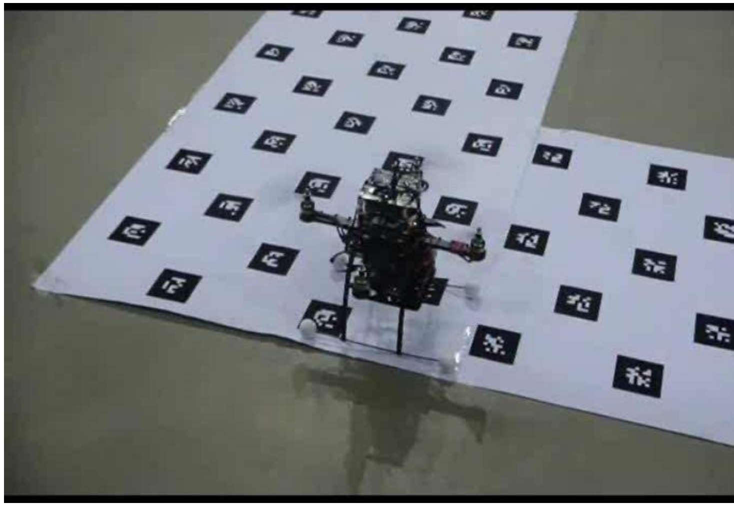
Application of calibration from planar grids

- Today, there are thousands of application of this algorithm:
  - Augmented reality
  - Robotics (beacon-based localization)
- Do we need to know the metric size of the tag?
  - For Augmented Reality?
  - For Robotics?





My lab

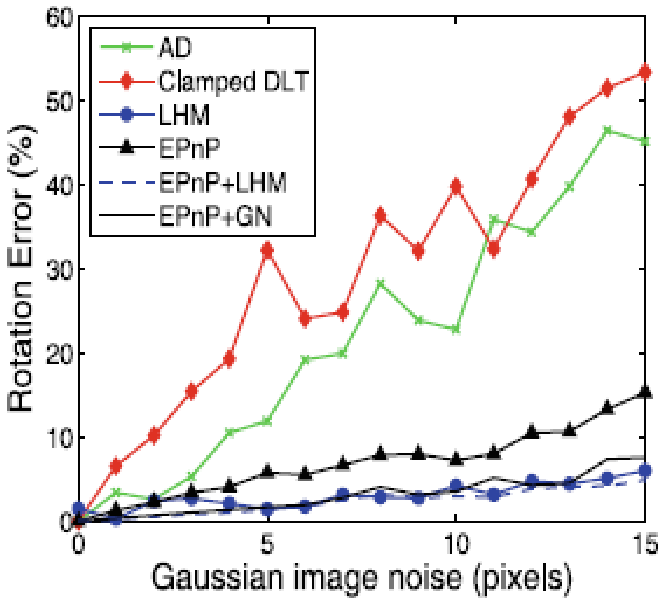


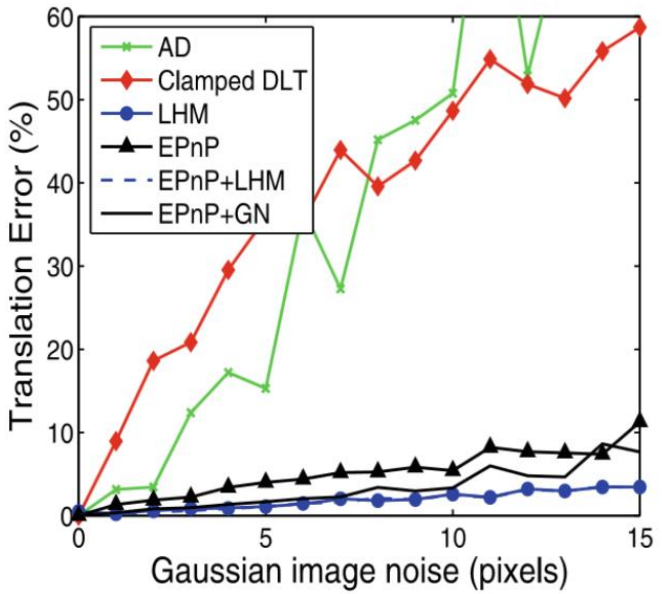
Marc Pollefeys' lab

Most AR apps use AprilTag or ARuco Markers

DLT vs PnP: Accuracy vs noise

If the camera is calibrated, only R and T need to be determined. In this case, should we use DLT (linear system of equations) or PnP (non linear)?

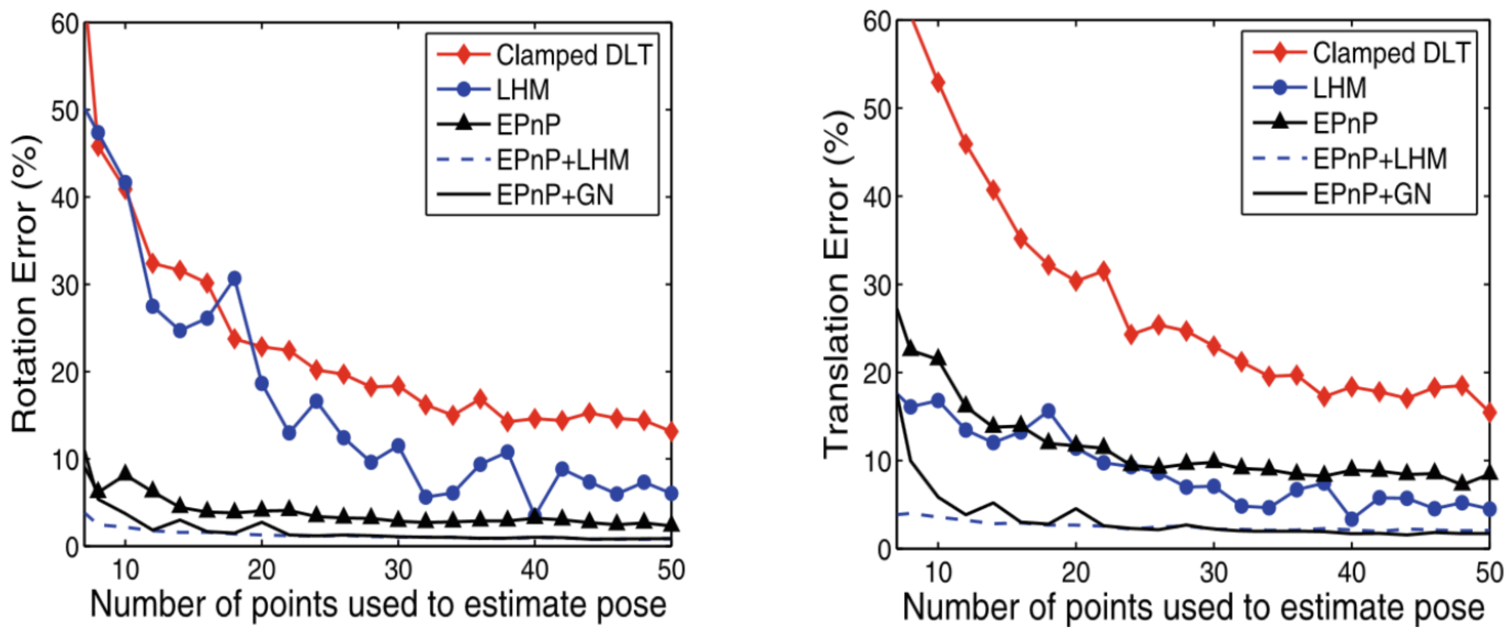




Lepetit, Moreno Noguer, Fua, EP*n*P: An Accurate *O(n)* Solution to the P*n*P Problem, IJCV'09

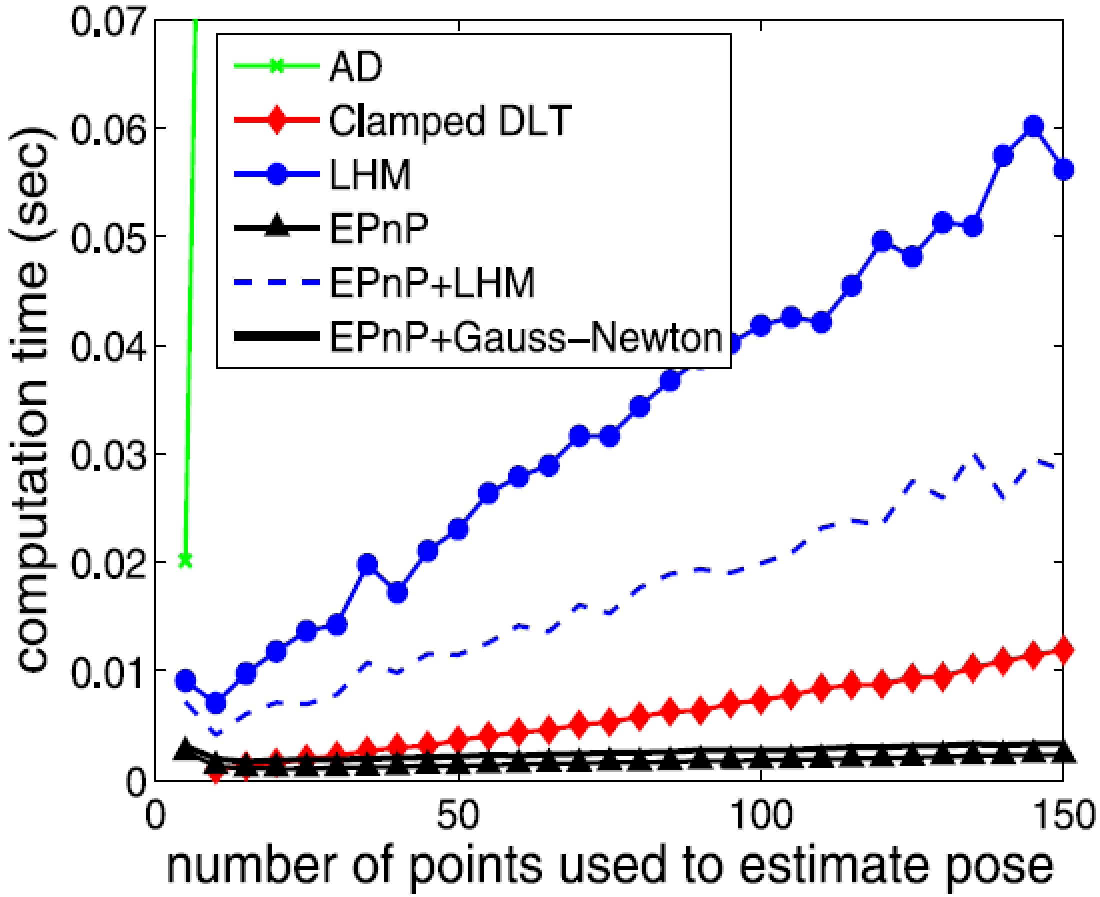
DLT vs PnP: Accuracy vs number of points

If the camera is calibrated, only R and T need to be determined. In this case, should we use DLT (linear system of equations) or PnP (non linear)?



Lepetit, Moreno Noguer, Fua, EP*n*P: An Accurate *O(n)* Solution to the P*n*P Problem, IJCV'09

DLT vs PnP: Timing



Lepetit, Moreno Noguer, Fua, EP*n*P: An Accurate *O(n)* Solution to the P*n*P Problem, IJCV'09

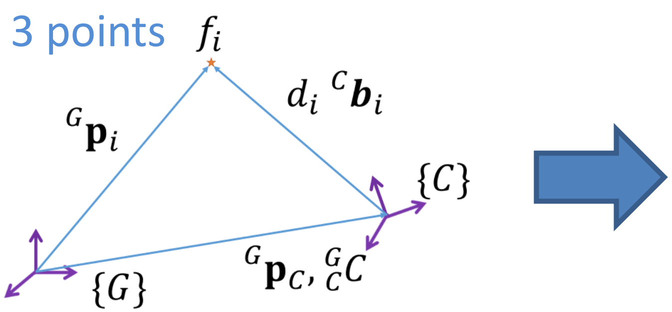
Won't be asked at the exam

An Efficient Algebraic Solution to P3P

Ke and Roumeliotis (CVPR'17)

Similarly to Kneip (CVPR'11) and Maselli (ICPR'14), directly solve for the camera's pose (not the distances).

1. Eliminate the camera's position and the features' distances to yield a system of 3 equations *in the camera's orientation alone*.



Pairwise subtraction and dot product

$$({}^G\mathbf{p}_1 - {}^G\mathbf{p}_2)^T\, {}^G_C\mathbf{C}({}^C\mathbf{b}_1 \times {}^C\mathbf{b}_2) = 0$$
$$({}^G\mathbf{p}_1 - {}^G\mathbf{p}_3)^T\, {}^G_C\mathbf{C}({}^C\mathbf{b}_1 \times {}^C\mathbf{b}_3) = 0$$
$$({}^G\mathbf{p}_2 - {}^G\mathbf{p}_3)^T\, {}^G_C\mathbf{C}({}^C\mathbf{b}_2 \times {}^C\mathbf{b}_3) = 0$$

2. Successively eliminate two of the unknown 3-DOFs (angles) algebraically and arrive at a *quartic polynomial equation*.

- **Results**: outperforms previous methods in terms of speed, accuracy, and robustness to close-to-singular cases.
- Available in **OpenCV > 3.3**. solvePnP() with SOLVEPNP_AP3P

Recap

PnP Problem

Calibrated camera **(i.e., instrinc parameters are known)**	**Uncalibrated camera** **(i.e., intrinsic parameters unknown)**
Works for **any 3D point configuration**	**DLT** (Direct Linear Transform)
Minimum number of points: **4** (i.e., 3+1) **P3P** (Perspective from Three Points (plus 1))	Minimum number of points: **4 if coplanar** **6 if non coplanar**

Outline of this lecture

- (Geometric) Camera calibration
  - PnP problem
    - P3P for **calibrated** cameras
    - DLT for **uncalibrated** cameras
      - From general 3D objects
      - From planar grids
- Omnidirectional cameras

Overview on Omnidirectional Cameras

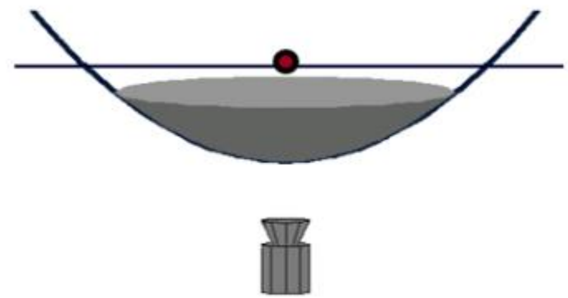
Fisheye

FOV > 130º

Wide FOV dioptric cameras (e.g. fisheye)

Catadioptric

360º all around

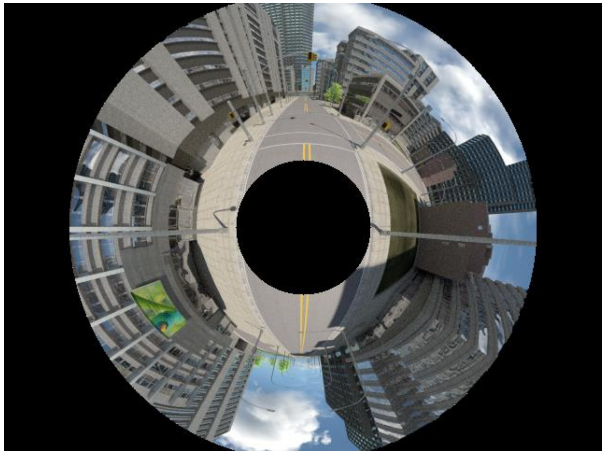


Catadioptric cameras (e.g. cameras and mirror systems)

Example scene viewed by three different camera models

Perspective

Fisheye



Catadioptric

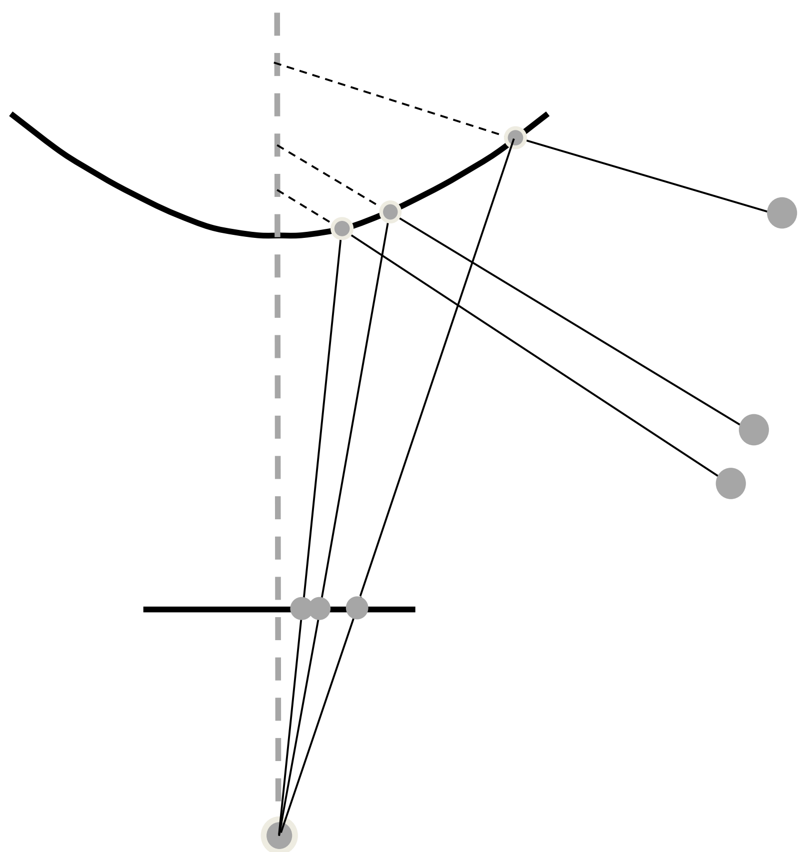
Zhang et al., Benefit of Large Field-of-View Cameras for Visual Odometry, ICRA’16
http://rpg.ifi.uzh.ch/fov.html

Catadioptric Cameras

Catadioptric Cameras

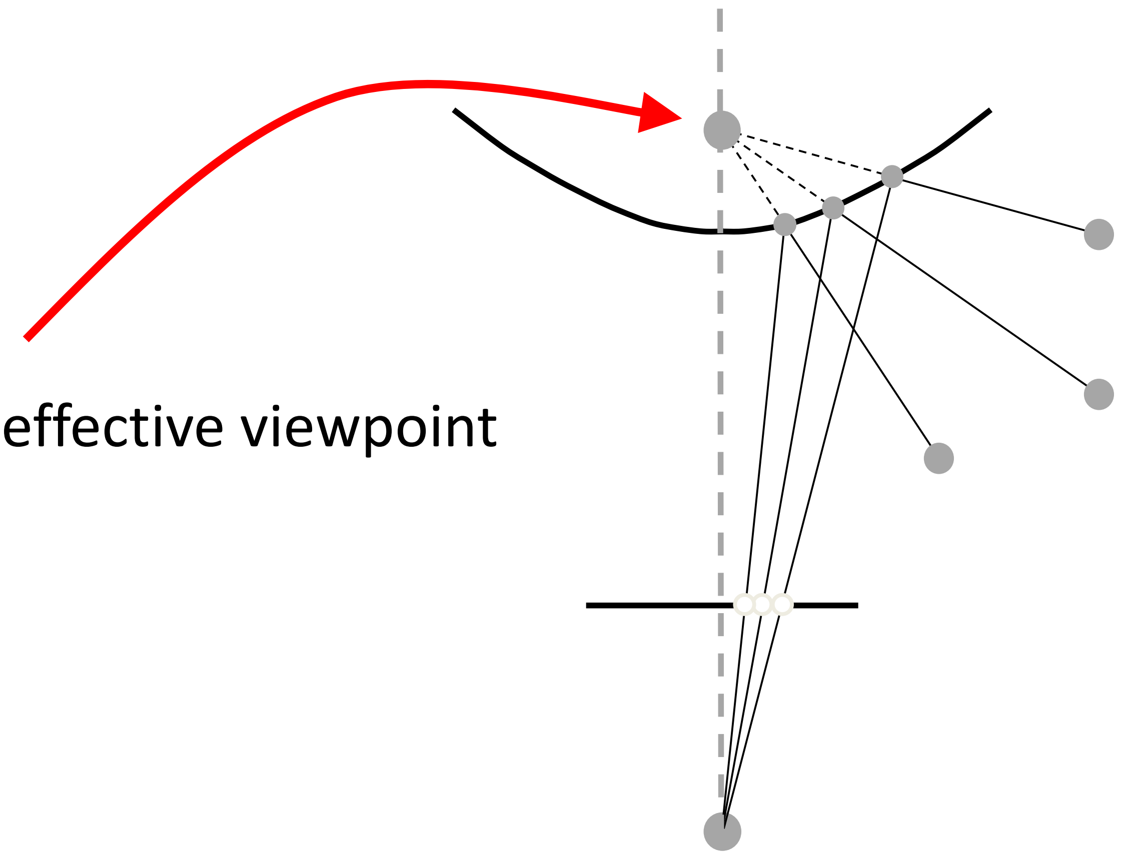
Non Central Catadioptric cameras

Rays do not intersect in a single point



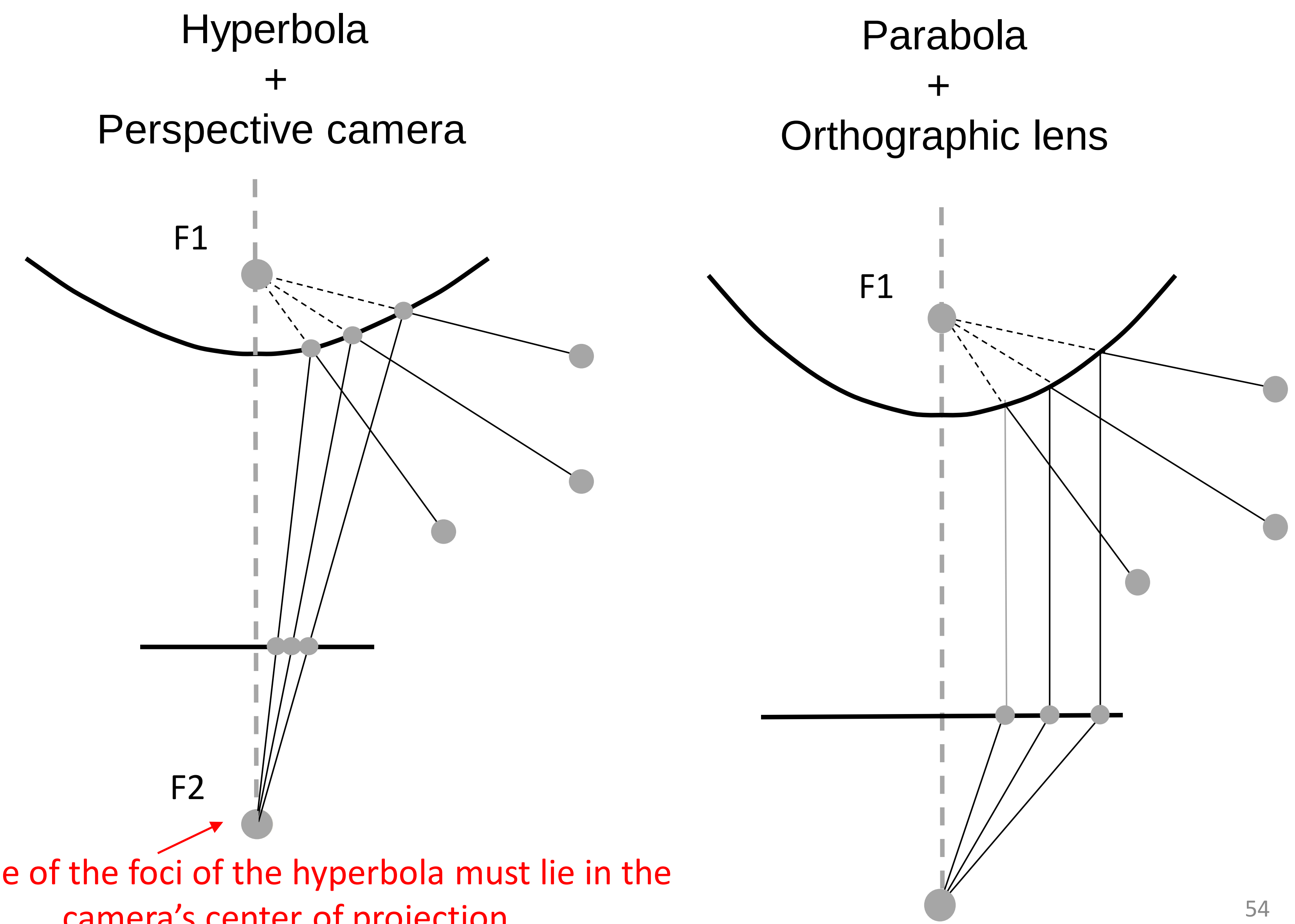
Central Catadioptric cameras

- Rays do not intersect in a single point
- Mirror must be **surface of revolution of a conic**



single effective viewpoint

Central Catadioptric cameras

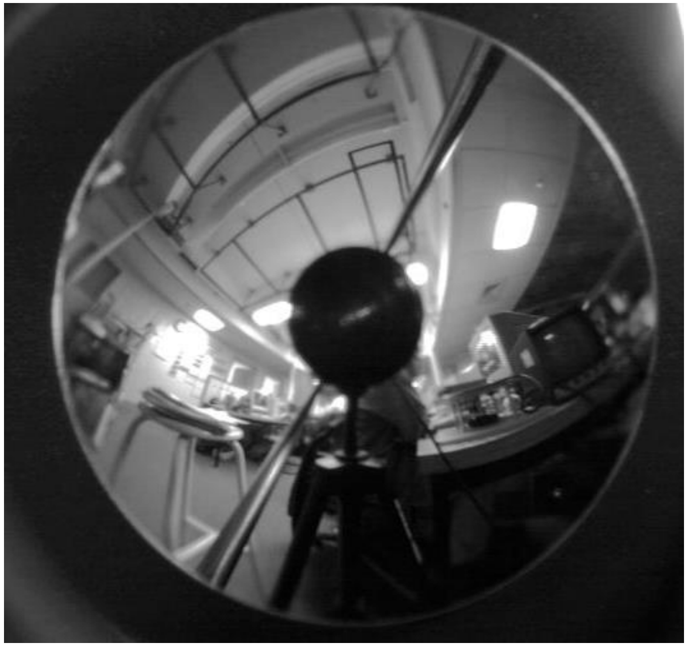


NB: one of the foci of the hyperbola must lie in the camera's center of projection

Why do we prefer central cameras?

Because we can:

- Apply standard algorithms valid for perspective geometry.
- Unwarp parts of an image into a perspective one
- Transform image points into normalized vectors on the unit sphere



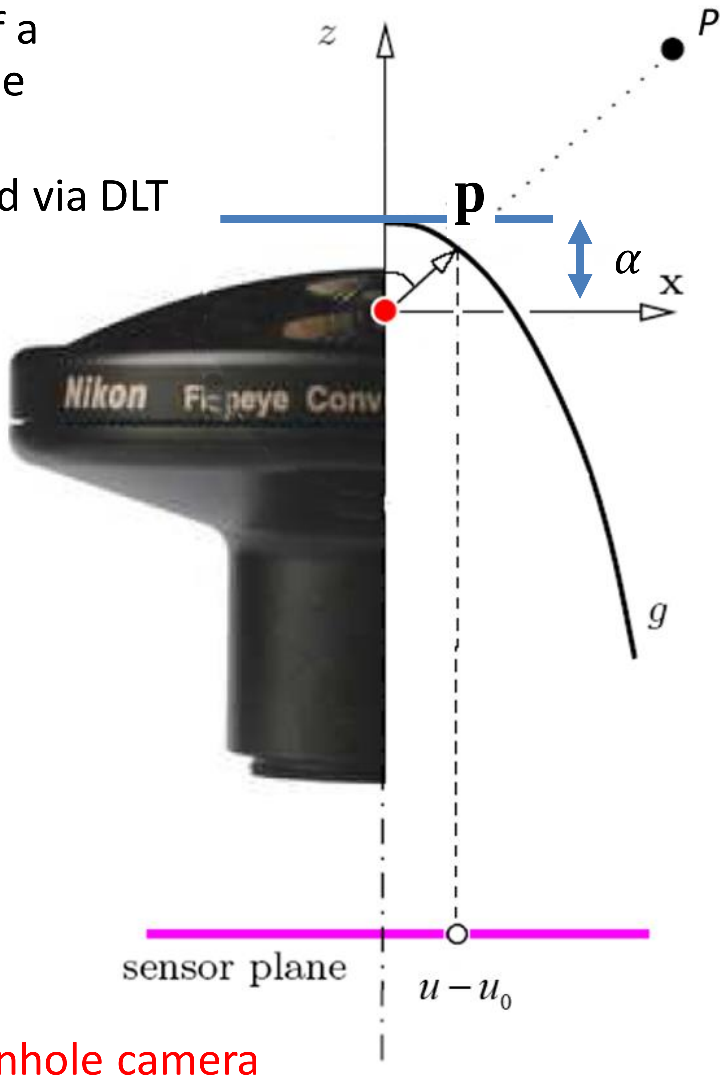
Unified Omnidirectional Camera Model

- We describe the *image projection function* by means of a polynomial, whose coefficients are the parameters to be estimated
- The coefficients, intrinsics, and extrinsics are then found via DLT

$$\lambda \cdot \mathbf{p} = \frac{\lambda}{\alpha} \cdot \begin{bmatrix} u - u_0 \\ v - v_0 \\ g(\rho) \end{bmatrix} = \begin{bmatrix} R & | & T \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

$$g(\rho) = \alpha + a_1\rho + a_2\rho^2 + \ldots + a_N\rho^N$$

$$\rho = \sqrt{(u-u_0)^2 + (v-v_0)^2}$$



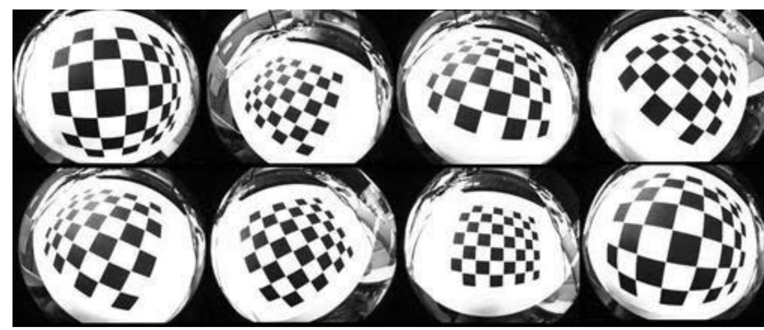
When $a_i = 0$ then we get a pinhole camera

Scaramuzza, Martinelli, Siegwart, *A toolbox for easily calibrating omnidirectional cameras*, IROS'06. PDF
Scaramuzza, *Omnidirectional Camera*, chapter of Encyclopedia of Computer Vision, Springer'14. PDF

OcamCalib: Omnidirectional Camera Calibration

- Released in 2006, it is the standard toolbox for calibrating wide angle cameras
- Original url: https://sites.google.com/site/scarabotix/ocamcalib-toolbox
- Since 2015, included in the Matlab Computer Vision Toolbox: https://ch.mathworks.com/help/vision/ug/fisheye-calibration-basics.html

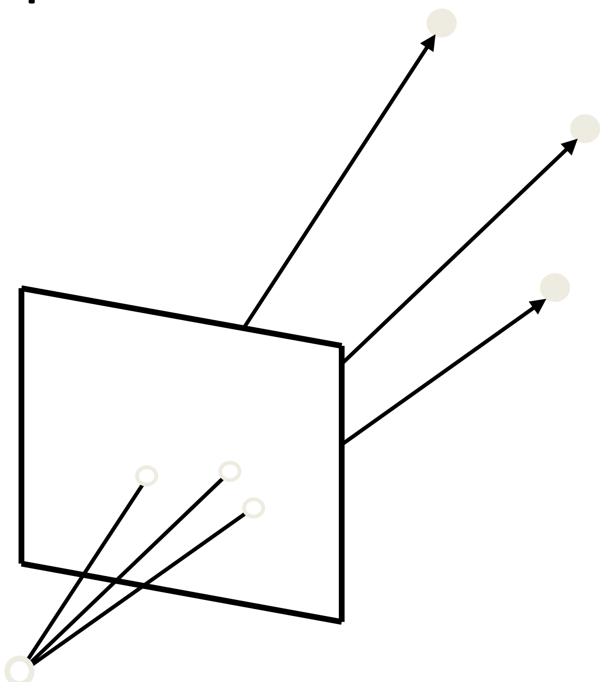
Example calibration images of a catadioptric camera

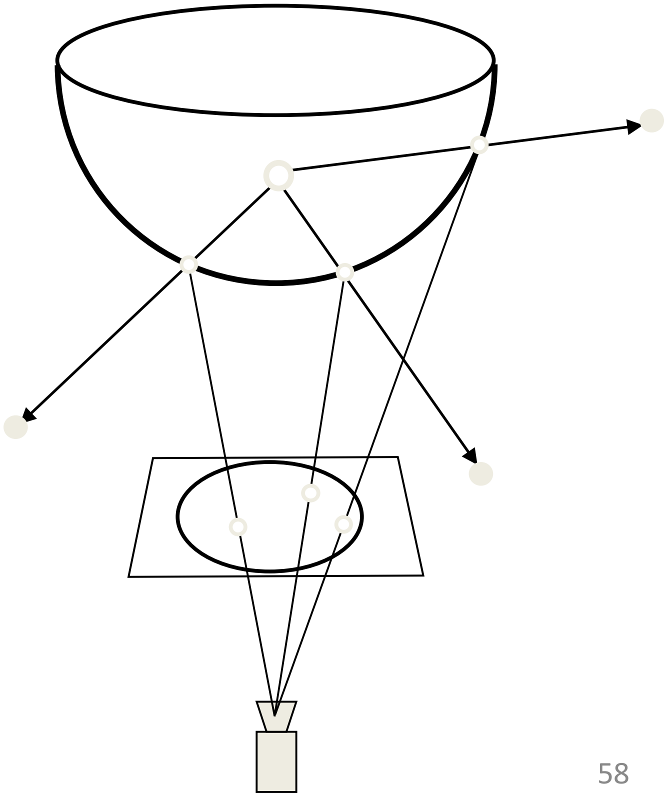


Example calibration images of a fisheye camera

D. Scaramuzza, A. Martinelli, R. Siegwart, A toolbox for easily calibrating omnidirectional cameras, IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006. PDF.

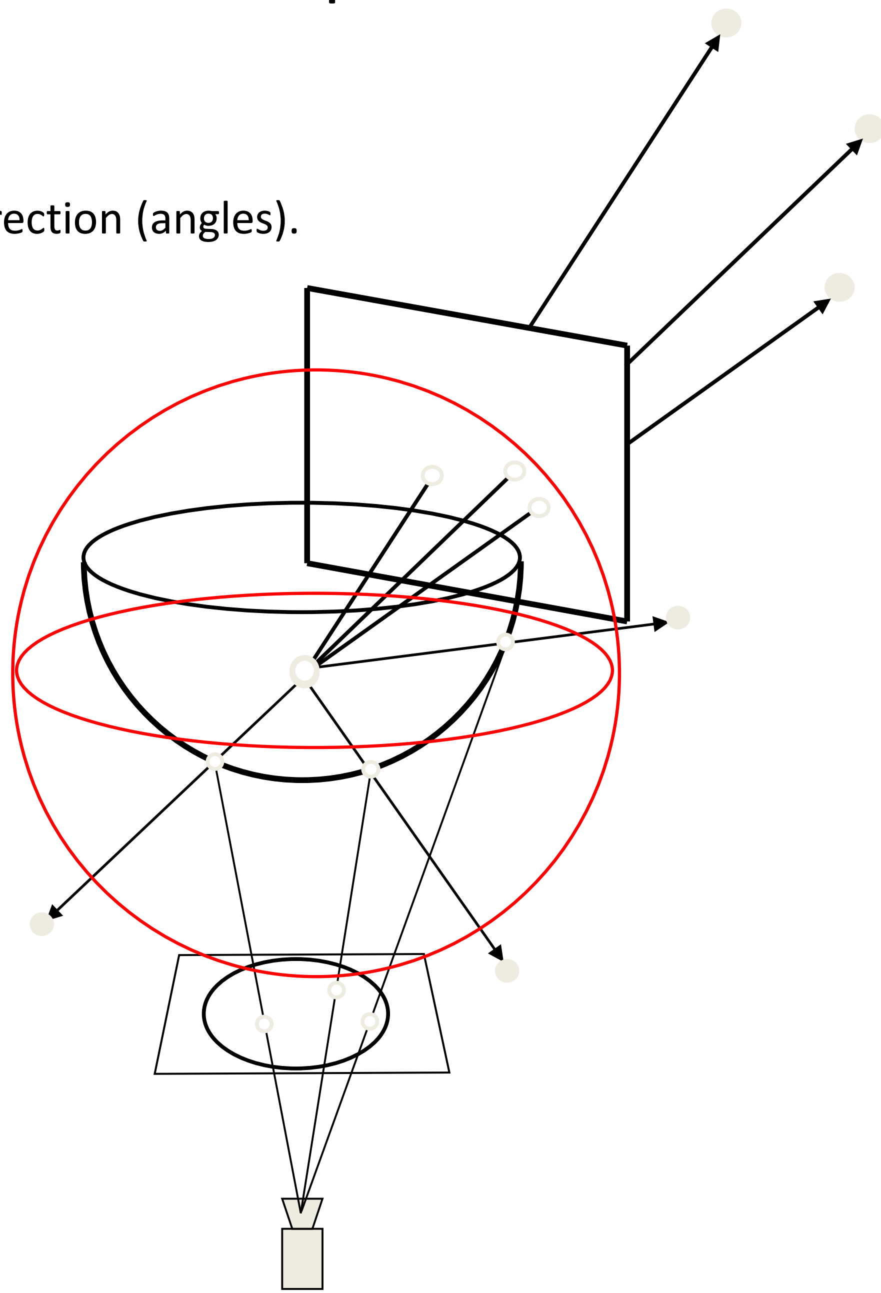
Equivalence between Perspective and Omnidirectional model





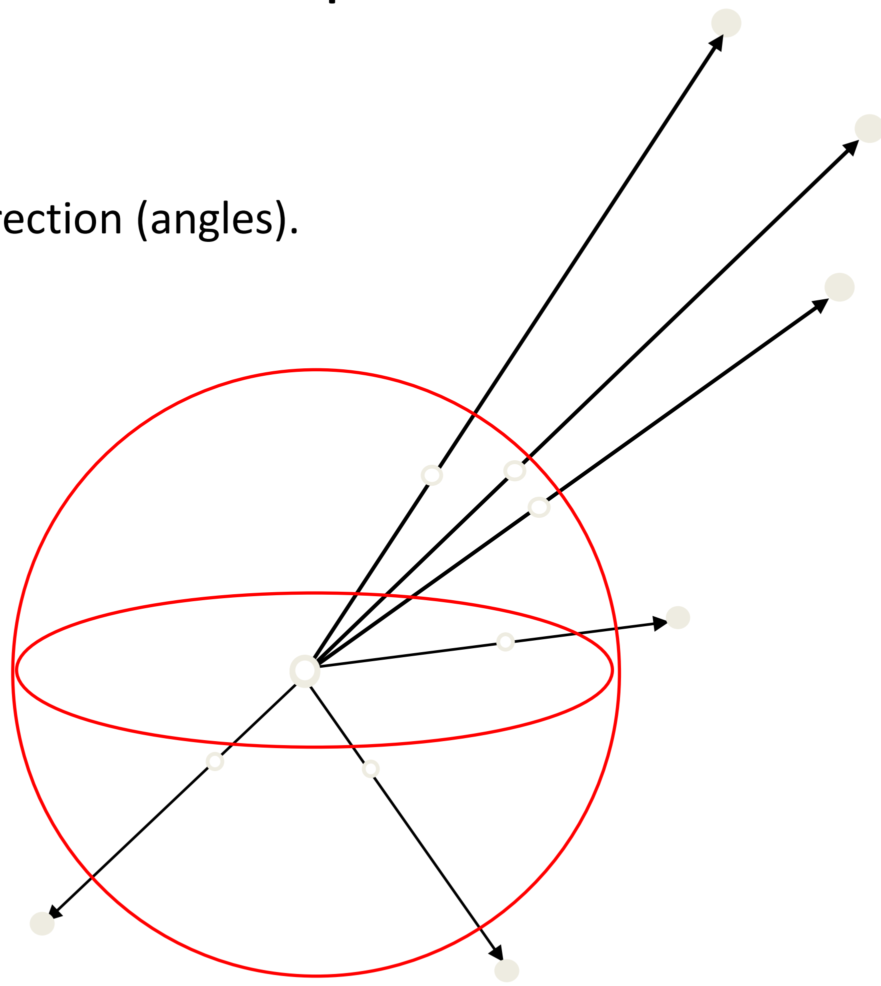
Equivalence between Perspective and Omnidirectional model

Measures the ray direction (angles).

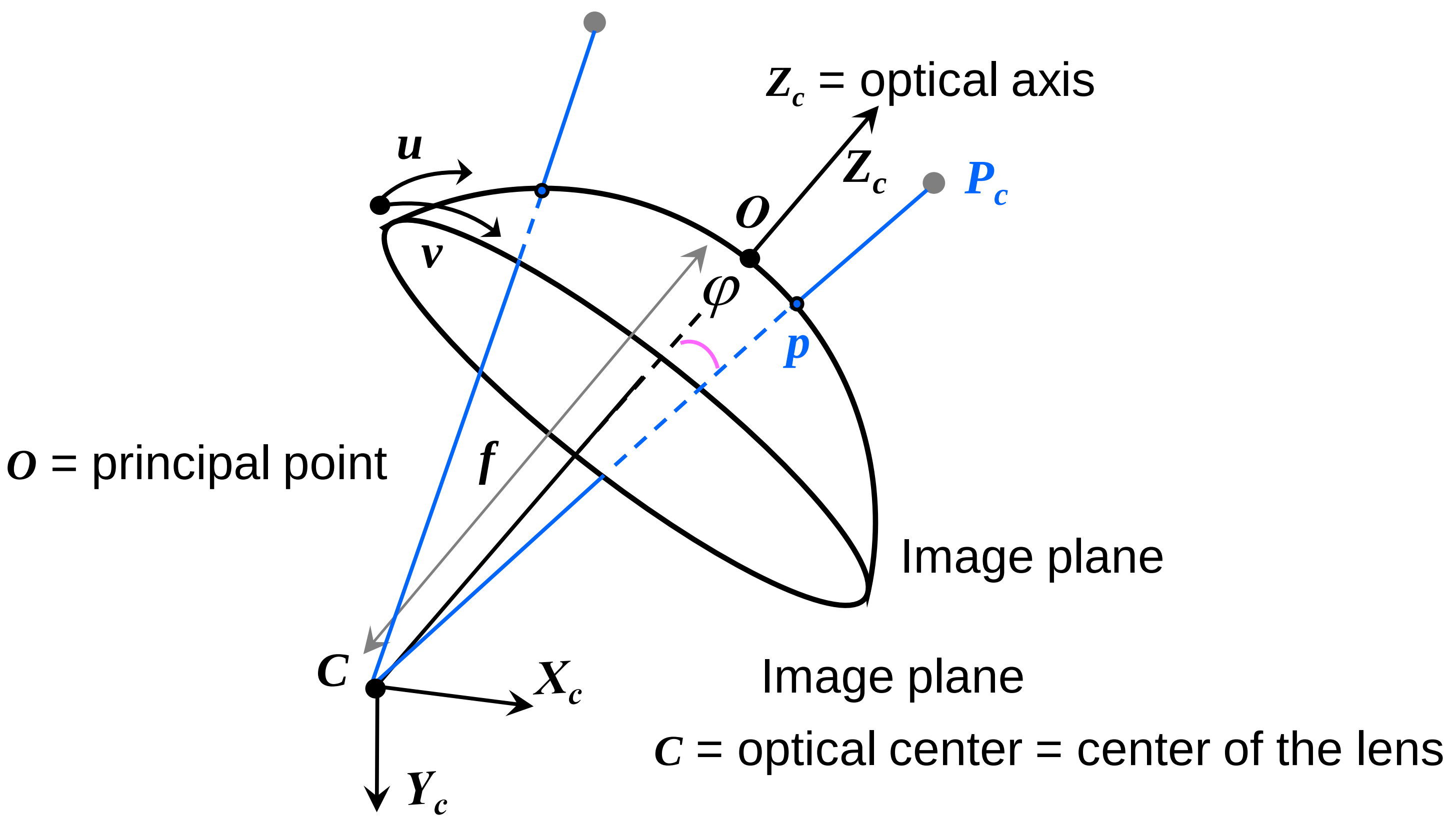


Equivalence between Perspective and Omnidirectional model

Measures the ray direction (angles).

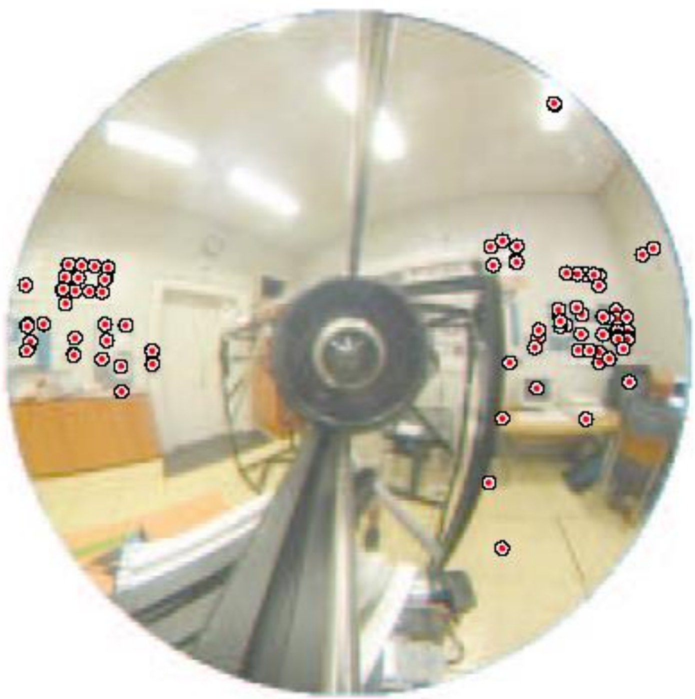


Equivalence between Perspective and Omnidirectional model

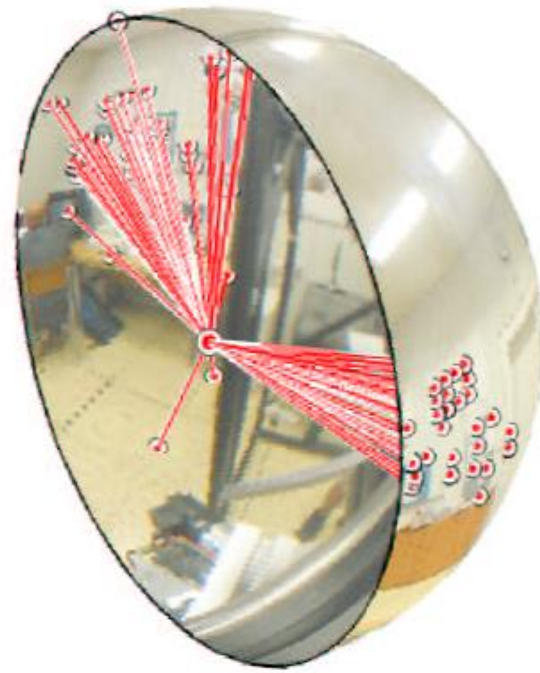


Representation of image points on the unit sphere

Always possible after the camera has been calibrated!



Points



Rays

Summary (things to remember)

- P3P and PnP problems
- DLT algorithm
- Calibration from planar grid (Homography algorithm)
- Readings: Chapter 2.1 of Szeliski book

- Omnidirectional cameras
  - Central and non central projection
  - Dioptric
  - Catadioptric (working principle of conic mirrors)
- Unified (spherical) model for perspective and omnidirectional cameras
- Reading: Chapter 4 of Autonomous Mobile Robots book: link

Understanding Check

Are you able to:

- Describe the general PnP problem and derive the behavior of its solutions?
- Explain the working principle of the P3P algorithm?
- Explain and derive the DLT? What is the minimum number of point correspondences it requires?
- Define central and non central omnidirectional cameras?
- What kind of mirrors ensure central projection?