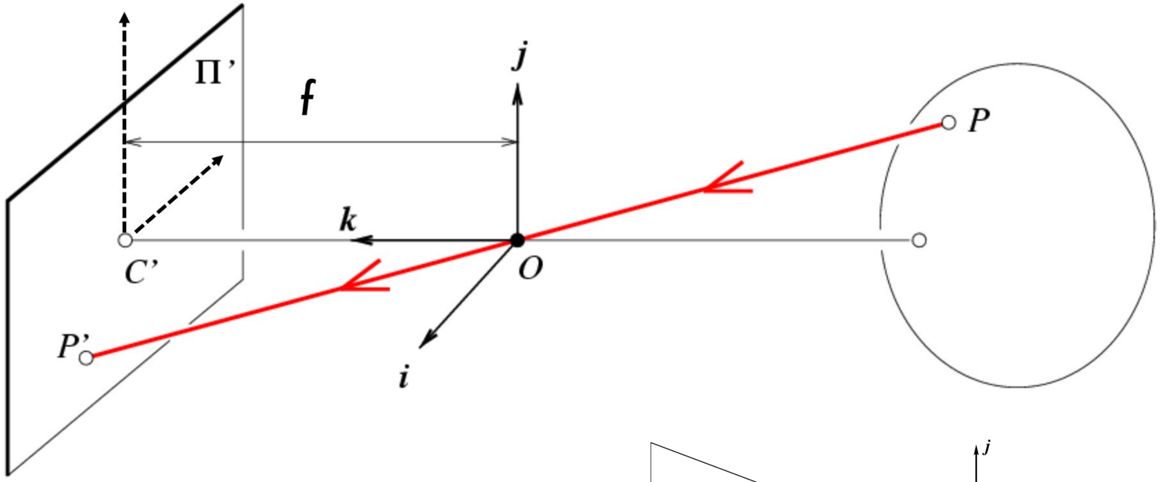
Camera Calibration and Pose Estimation

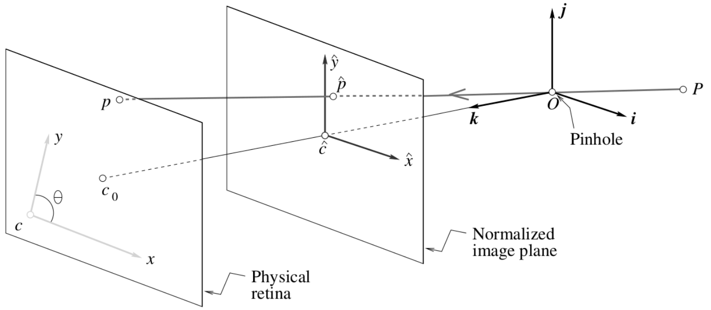
CS 6384 Computer Vision
Professor Yu Xiang
The University of Texas at Dallas

Some slides of this lecture are courtesy Silvio Savarese

Recap Camera Models

- Camera projection matrix





$$P = K[R|\mathbf{t}]$$

3x3 3x4

Camera intrinsics

Camera extrinsics: rotation and translation

$$K = \begin{bmatrix} \alpha & -\alpha \cot\theta & c_x \\ 0 & \frac{\beta}{\sin\theta} & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Camera Calibration

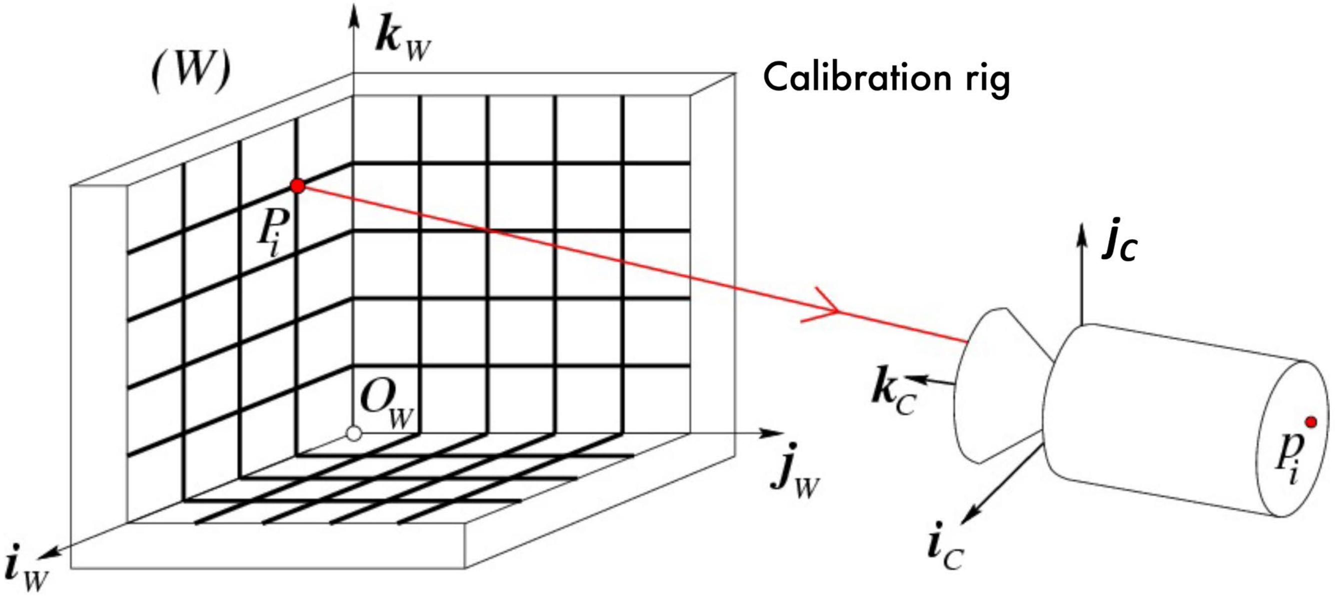
- Estimate the camera intrinsics and camera extrinsics $P = K[R|\mathbf{t}]$

- Why is this useful?
  - If we know K and depth, we can compute 3D points in camera frame
  - In stereo matching to compute depth, we need to know focal length
  - Camera pose tracking is critical in SLAM (Simultaneous Localization and Mapping)

Camera Calibration

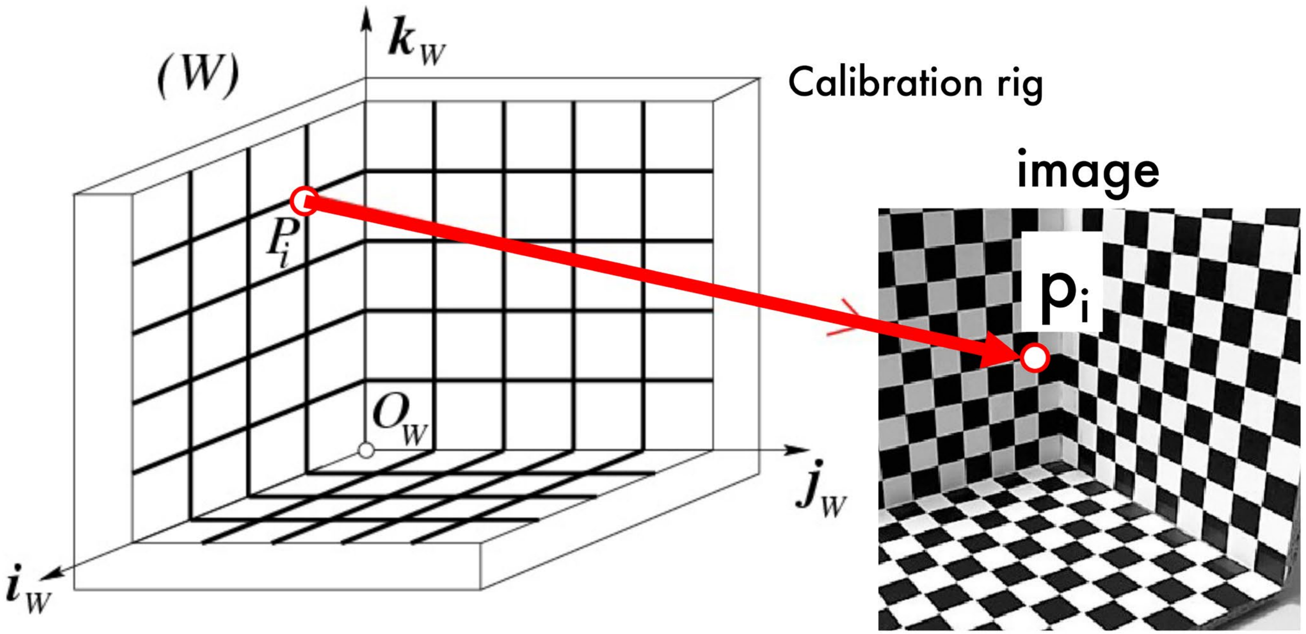
- Estimate the camera intrinsics and camera extrinsics $P = K[R|\mathbf{t}]$

- Idea: using images from the camera with a known world coordinate frame



checkerboard

Camera Calibration



- Unknowns

Camera intrinsics K

Camera extrinsics: rotation and translation R, T

- Knowns

World coordinates $P_1, \ldots, P_n$

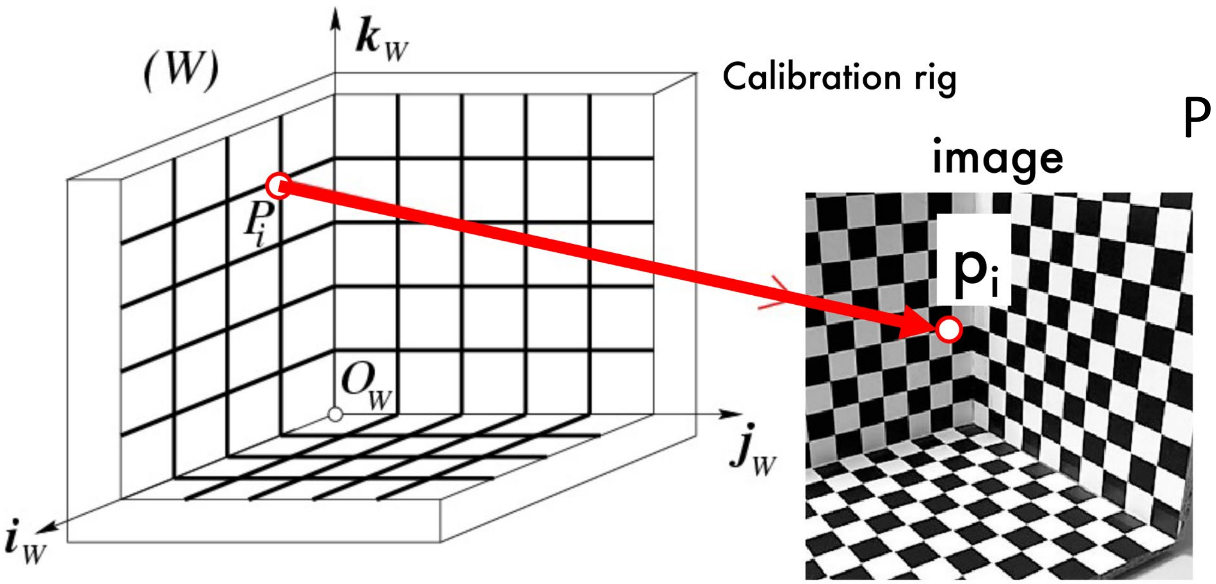
Pixel coordinates $p_1, \ldots, p_n$

Camera Calibration

$$K = \begin{bmatrix} \alpha & -\alpha \cot \theta & c_x \\ 0 & \frac{\beta}{\sin \theta} & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$p_i = MP_i = K[R|T]P_i$$

Pixel coordinate   3x4   World coordinate



- How many unknowns in M?
  - 11
- How many correspondences do we need to estimate M?
  - We need 11 equations
  - 6 correspondences
- More correspondences are better

A Linear Approach to Camera Calibration

$$p_i = MP_i = K[R|T]P_i$$

$$M = \begin{bmatrix} \mathbf{m}_1 \\ \mathbf{m}_2 \\ \mathbf{m}_3 \end{bmatrix} \begin{array}{c} 1 \times 4 \\ 1 \times 4 \\ 1 \times 4 \end{array} \quad MP_i = \begin{bmatrix} \mathbf{m}_1 P_i \\ \mathbf{m}_2 P_i \\ \mathbf{m}_3 P_i \end{bmatrix} \quad \underset{\text{Pixel}}{p_i} = \begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} \frac{\mathbf{m}_1 P_i}{\mathbf{m}_3 P_i} \\ \frac{\mathbf{m}_2 P_i}{\mathbf{m}_3 P_i} \end{bmatrix}$$

A pair of equations

$$u_i(m_3 P_i) - m_1 P_i = 0$$
$$v_i(m_3 P_i) - m_2 P_i = 0$$

A Linear Approach to Camera Calibration

- Given n correspondences $p_i = \begin{bmatrix} u_i \\ v_i \end{bmatrix} \leftrightarrow P_i$

2n equations

$$\begin{aligned} u_1(m_3 P_1) - m_1 P_1 &= 0 \\ v_1(m_3 P_1) - m_2 P_1 &= 0 \\ &\vdots \\ u_n(m_3 P_n) - m_1 P_n &= 0 \\ v_n(m_3 P_n) - m_2 P_n &= 0 \end{aligned}$$

$$\begin{bmatrix} P_1^T & 0^T & -u_1 P_1^T \\ 0^T & P_1^T & -v_1 P_1^T \\ & \vdots & \\ P_n^T & 0^T & -u_n P_n^T \\ 0^T & P_n^T & -v_n P_n^T \end{bmatrix} \begin{bmatrix} m_1^T \\ m_2^T \\ m_3^T \end{bmatrix} = \mathbf{P} m = 0$$

$2n \times 12 \qquad 12 \times 1$

How to solve this linear system?

Linear System

$$\mathbf{P}m = 0$$

$$2n \times 12 \quad 12 \times 1$$

- Find non-zero solutions
- If m is a solution, k×m is also a solution for $k \in \mathcal{R}$
- We can seek a solution $||m|| = 1$

$$\min ||\mathbf{P}m||$$

Subject to $||m|| = 1$

Solution: $P = UDV^T$ SVD decomposition of P

$$2n \times 12 \quad 12 \times 12 \quad 12 \times 12$$

m is the last column of V

A5.3 in Multiview Geometry in Computer Vision

A Linear Approach to Camera Calibration

$$p_i = MP_i = K[R|T]P_i$$

$$\mathbf{P}m = 0$$

m is the last column of V

$$m \to M$$ Up to scale

How to extract K, R and T from M?

$$K = \begin{bmatrix} \alpha & -\alpha\cot\theta & c_x \\ 0 & \frac{\beta}{\sin\theta} & c_y \\ 0 & 0 & 1 \end{bmatrix} \qquad R = \begin{bmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \mathbf{r}_3^T \end{bmatrix} \qquad T = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

3 rows

$$\rho M = \begin{bmatrix} \alpha r_1^T - \alpha\cot\theta r_2^T + c_x r_3^T & \alpha t_x - \alpha\cot\theta t_y + c_x t_z \\ \frac{\beta}{\sin\theta} r_2^T + c_y r_3^T & \frac{\beta}{\sin\theta} t_y + c_y t_z \\ r_3^T & t_z \end{bmatrix}$$

Scale

A Linear Approach to Camera Calibration

$$M = \frac{1}{\rho}\begin{bmatrix} \alpha r_1^T - \alpha\cot\theta r_2^T + c_x r_3^T & \alpha t_x - \alpha\cot\theta t_y + c_x t_z \\ \frac{\beta}{\sin\theta} r_2^T + c_y r_3^T & \frac{\beta}{\sin\theta} t_y + c_y t_z \\ r_3^T & t_z \end{bmatrix} = \begin{bmatrix} A & b \end{bmatrix} = \begin{bmatrix} a_1^T \\ a_2^T \\ a_3^T \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

The rows of a rotation matrix are unit-length, perpendicular to each other

Intrinsics

$$K = \begin{bmatrix} \alpha & -\alpha\cot\theta & c_x \\ 0 & \frac{\beta}{\sin\theta} & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$\rho = \pm\frac{1}{\|a_3\|}$$

$$c_x = \rho^2(a_1 \cdot a_3)$$

$$c_y = \rho^2(a_2 \cdot a_3)$$

$$\theta = \cos^{-1}\left(-\frac{(a_1 \times a_3)\cdot(a_2 \times a_3)}{\|a_1 \times a_3\| \cdot \|a_2 \times a_3\|}\right)$$

$$\alpha = \rho^2\|a_1 \times a_3\| \sin\theta$$

$$\beta = \rho^2\|a_2 \times a_3\| \sin\theta$$

Extrinsics

$$r_1 = \frac{a_2 \times a_3}{\|a_2 \times a_3\|}$$

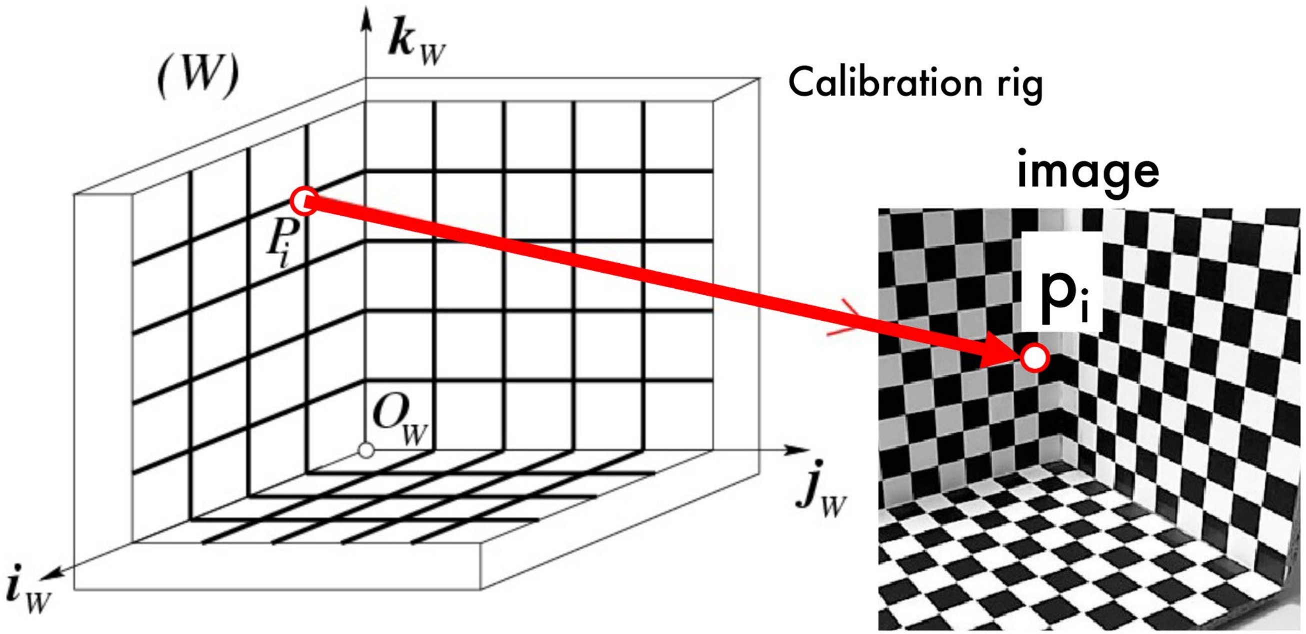
$$r_2 = r_3 \times r_1$$

$$r_3 = \rho a_3$$

$$T = \rho K^{-1} b$$

FP, Computer Vision: A Modern Approach, Sec. 3.2.2

A Linear Approach to Camera Calibration

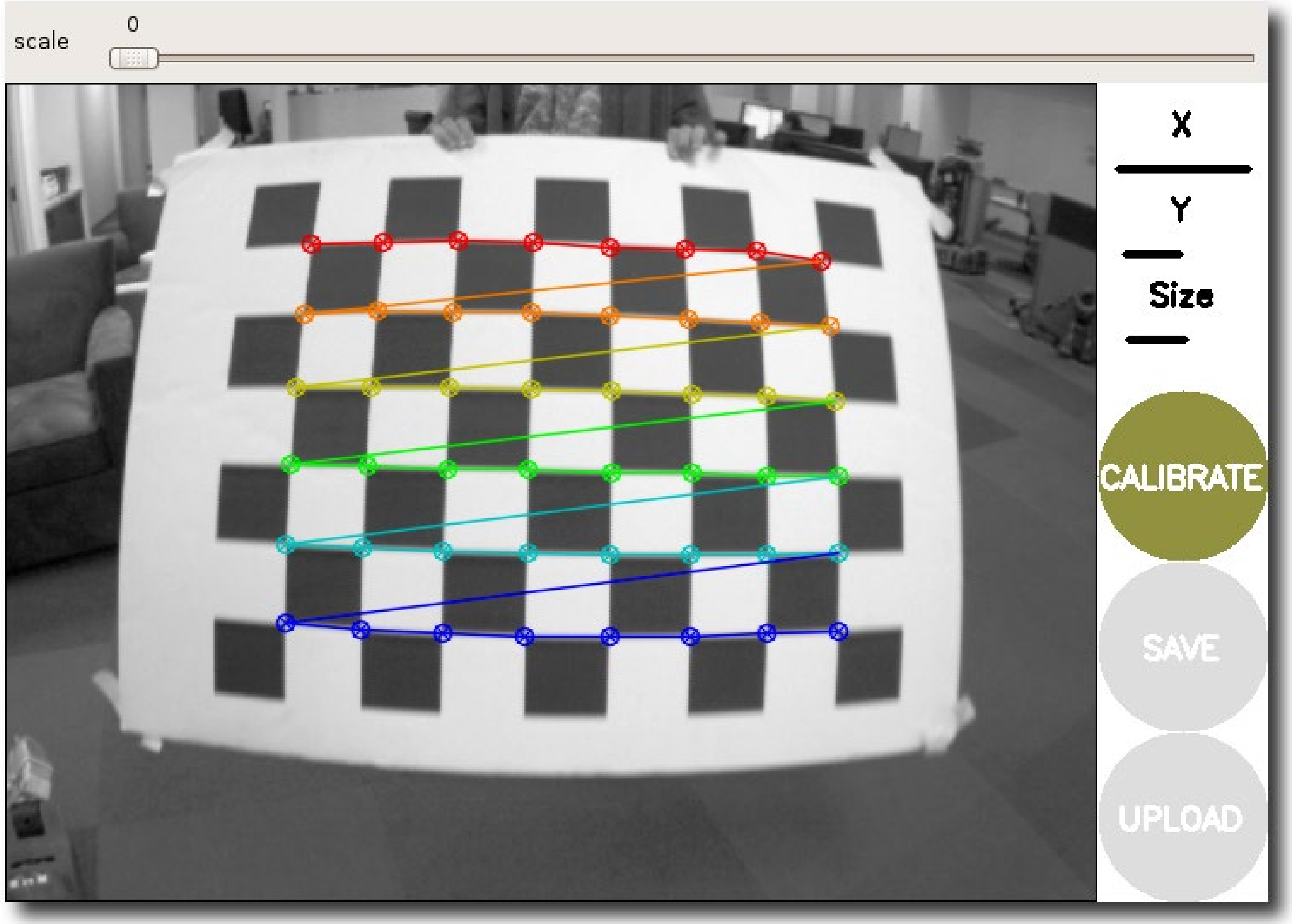


$$\mathbf{P}m = 0$$

All 3D points should NOT be on the same plane. Otherwise, no solution

FP, Computer Vision: A Modern Approach, Sec. 1.3

Camera Calibration with a 2D Plane



Harris Corner Detection

http://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration

Camera Calibration with a 2D Plane

- 3D point $P = \begin{bmatrix} X \\ Y \\ 0 \end{bmatrix}$
- Homography between the model plane and its image

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Scale factor

Pixel on the checkerboard

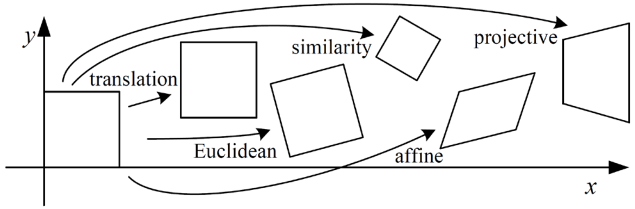
Intrinsics

Extrinsics

3D points on the checkerboard (Z = 0)

Homography 3×3

$$\mathbf{H} = \mathbf{A} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix}$$



A Flexible New Technique for Camera Calibration. Zhengyou Zhang, TPAMI, 2000.

Camera Calibration with a 2D Plane

- Homography between the model plane and its image
  - Given the correspondences, we can estimate H

$$\mathbf{H} = \mathbf{A}[\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{t}]$$

$$\mathbf{H} = [\mathbf{h}_1 \quad \mathbf{h}_2 \quad \mathbf{h}_3] \qquad [\mathbf{h}_1 \quad \mathbf{h}_2 \quad \mathbf{h}_3] = \lambda \mathbf{A}[\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{t}]$$

$\mathbf{r}_1 \ \mathbf{r}_2$ are orthogonal and with unit length

2 equations for intrinsics

$$\mathbf{h}_1^T \mathbf{A}^{-T} \mathbf{A}^{-1} \mathbf{h}_2 = 0$$

$$\mathbf{h}_1^T \mathbf{A}^{-T} \mathbf{A}^{-1} \mathbf{h}_1 = \mathbf{h}_2^T \mathbf{A}^{-T} \mathbf{A}^{-1} \mathbf{h}_2$$

Given n images, 2n equations for A

Solve the linear system for A

A Flexible New Technique for Camera Calibration. Zhengyou Zhang, TMAPI. 2000.

Camera Calibration with a 2D Plane

- Homography between the model plane and its image

$$[\mathbf{h}_1 \quad \mathbf{h}_2 \quad \mathbf{h}_3] = \lambda \mathbf{A} [\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{t}]$$

Extrinsics

$$\mathbf{r}_1 = \lambda \mathbf{A}^{-1}\mathbf{h}_1 \qquad \mathbf{r}_2 = \lambda \mathbf{A}^{-1}\mathbf{h}_2 \qquad \mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2 \qquad \mathbf{t} = \lambda \mathbf{A}^{-1}\mathbf{h}_3$$

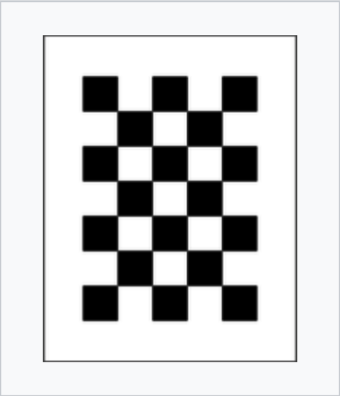
Afterwards, refine all the parameters including lens distortion parameters

$$\sum_{i=1}^{n} \sum_{j=1}^{m} \|\mathbf{m}_{ij} - \hat{\mathbf{m}}(\mathbf{A}, \mathbf{R}_i, \mathbf{t}_i, \mathbf{M}_j)\|^2$$

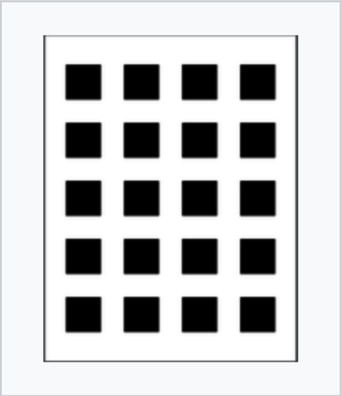
2D projection

A Flexible New Technique for Camera Calibration. Zhengyou Zhang, TPAMI, 2000.

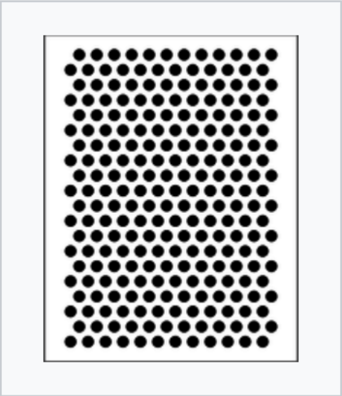
Calibration Patterns



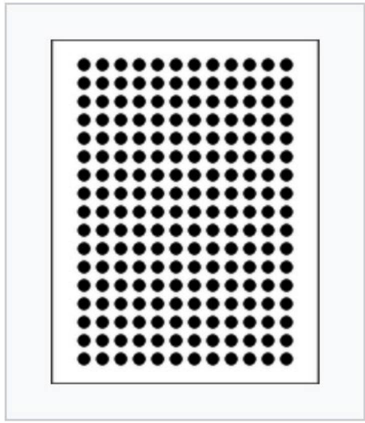
Chessboard



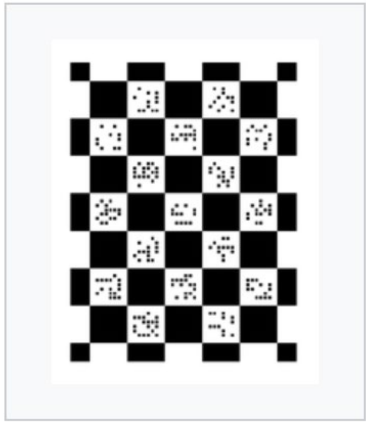
Square Grid



Circle Hexagonal Grid

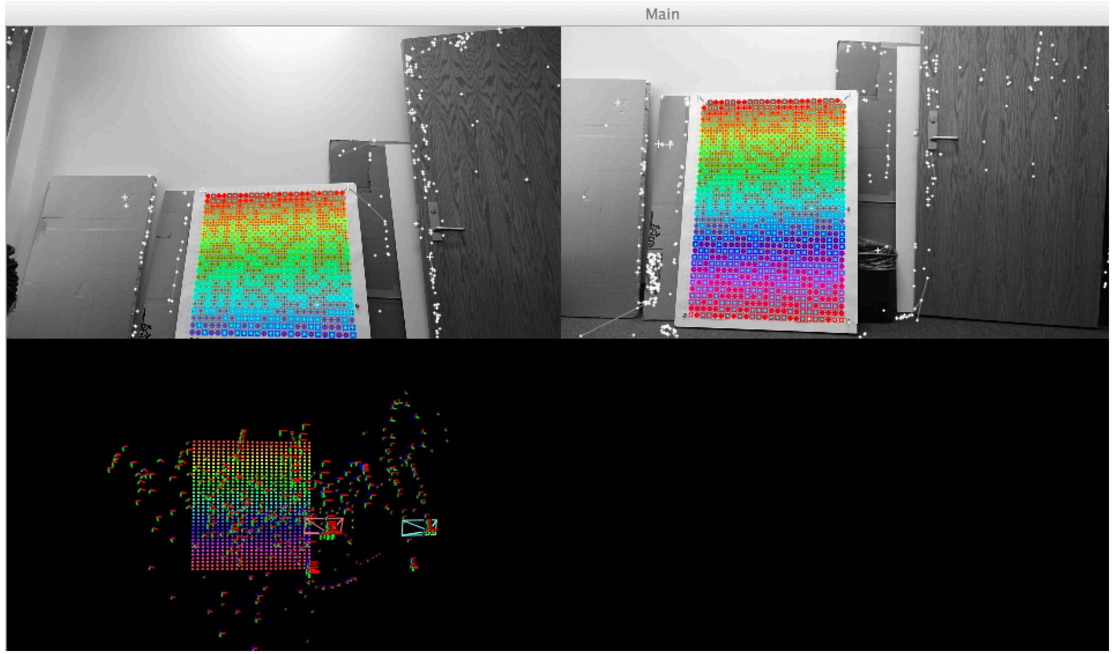


Circle Regular Grid



ECoCheck

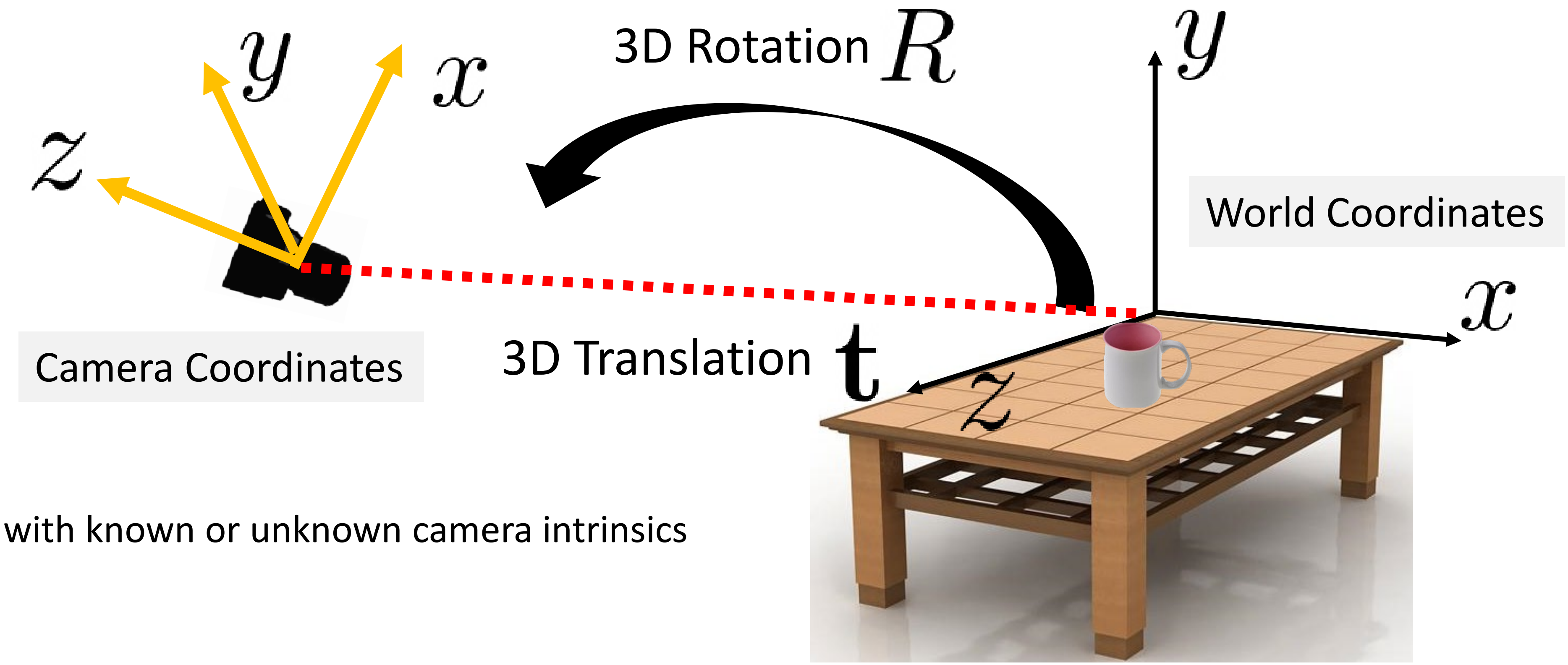
https://boofcv.org/index.php?title=Tutorial_Camera_Calibration



https://github.com/arpg/Documentation/tree/master/Calibration

Camera Pose Estimation

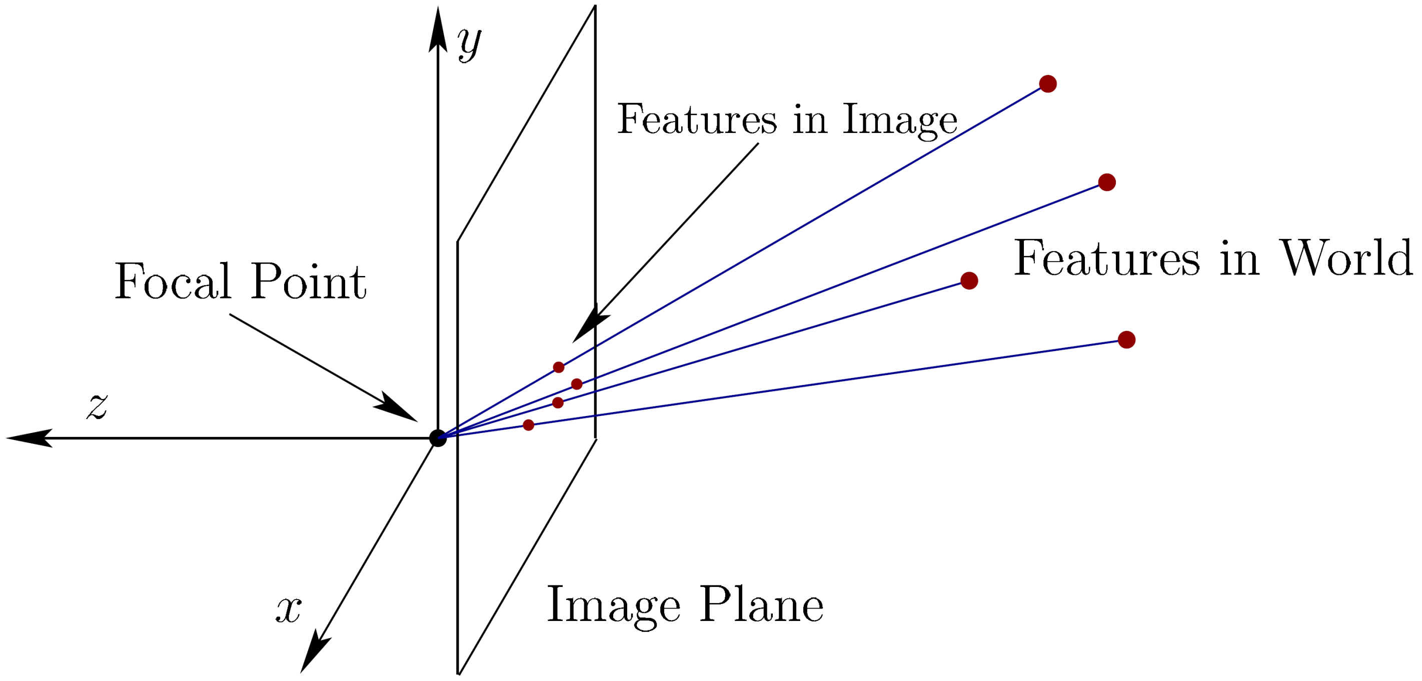
- Estimate the 3D rotation and 3D translation of a camera with respect to some world coordinate frame



Two cases: with known or unknown camera intrinsics

Camera Pose Estimation

- Using visibility of features in the real world

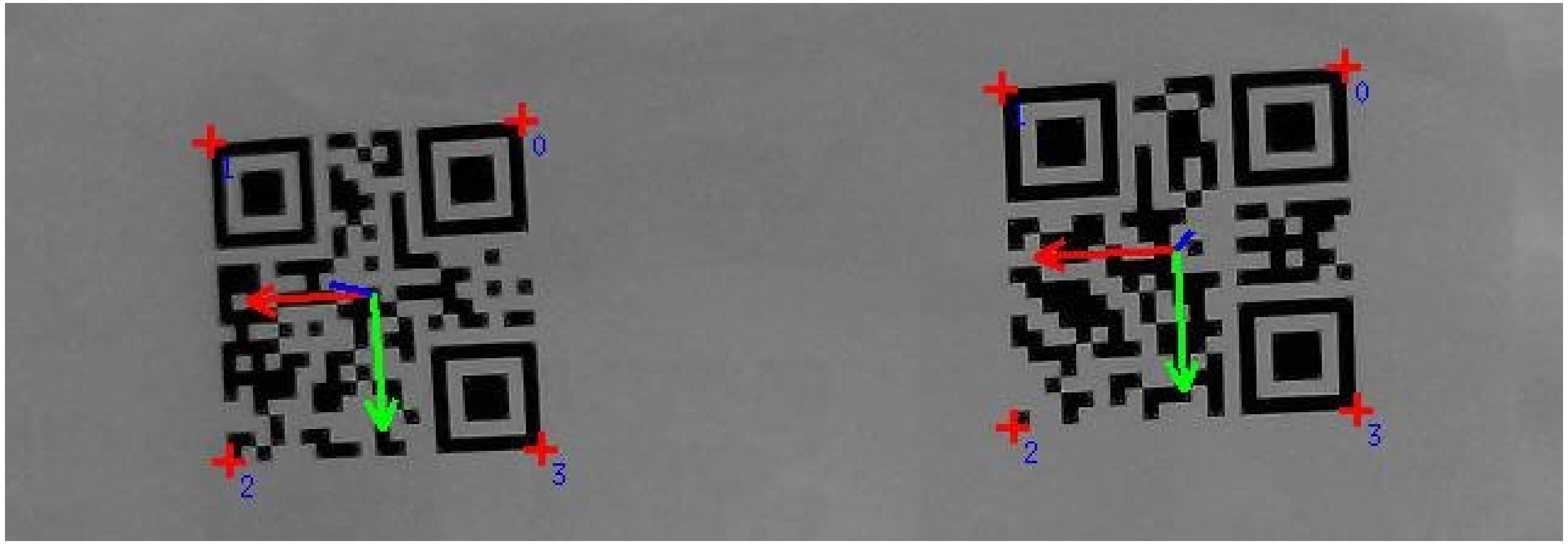


- Natural Features
  - No setup cost
  - A difficult problem
- Artificial features
  - Print a special tag

QR code

QR Code for Pose Estimation

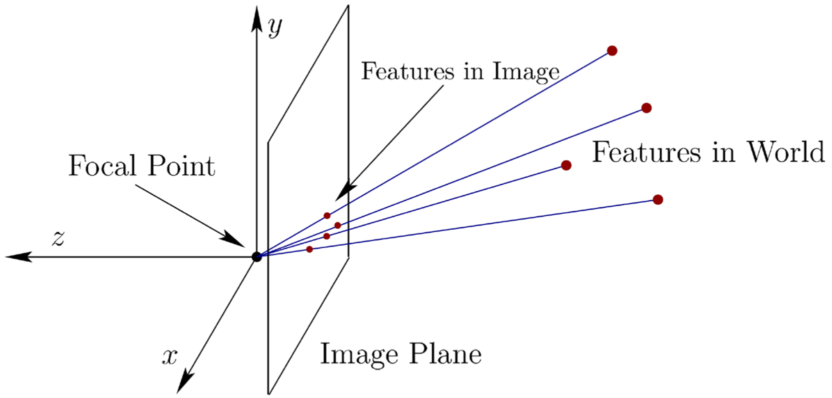
- Using the 4 corners of a QR code as features



https://visp-doc.inria.fr/doxygen/visp-daily/tutorial-pose-estimation-qrcode.html

The Perspective-n-Point (PnP) Problem

- Given/known variables
  - A set of n 3D points in the world coordinates p_w
  - Their projections (2D coordinates) on an image p_c
  - Camera intrinsics K
- Unknown variables
  - 3D rotation of the camera with respective to the world coordinates R
  - 3D translation of the camera T

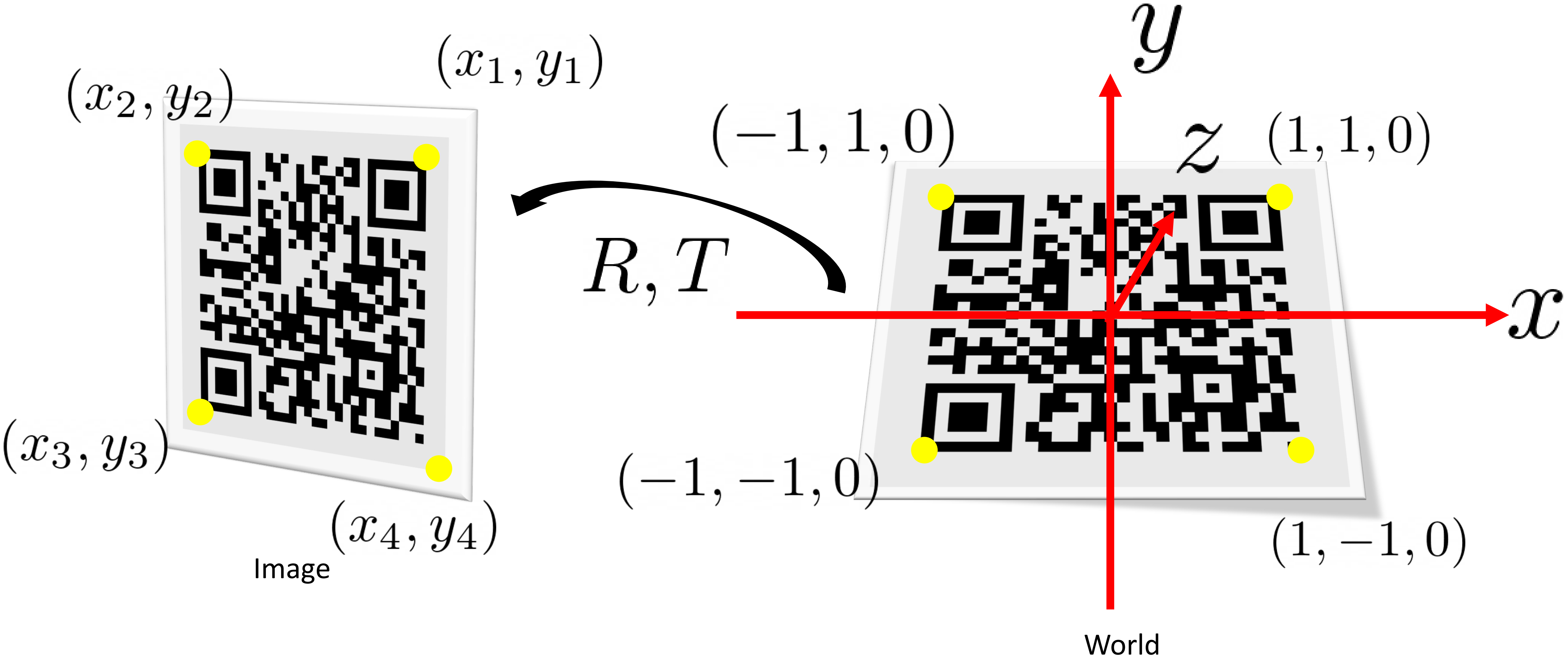


$$s\,p_c = K\,[\,R\,|\,T\,]\,p_w$$

Unknown

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

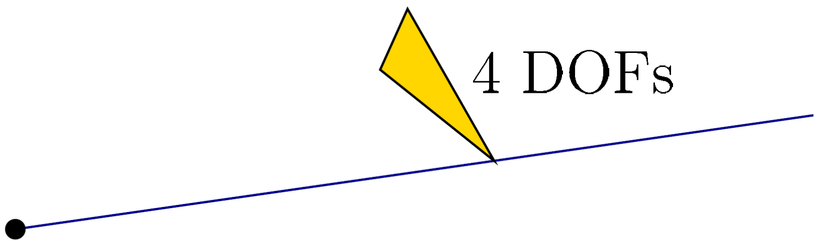
The PnP Problem with QR Code



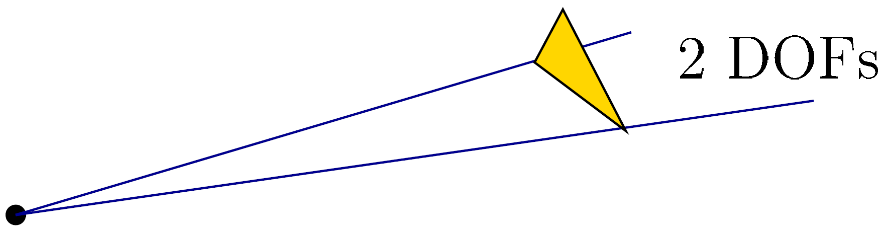
The Perspective-n-Point (PnP) Problem

- 6 degrees of freedom (DOFs)
  - 3 DOF rotation, 3 DOF translation

- Each feature that is visible eliminates 2 DOFs



P1P



P2P

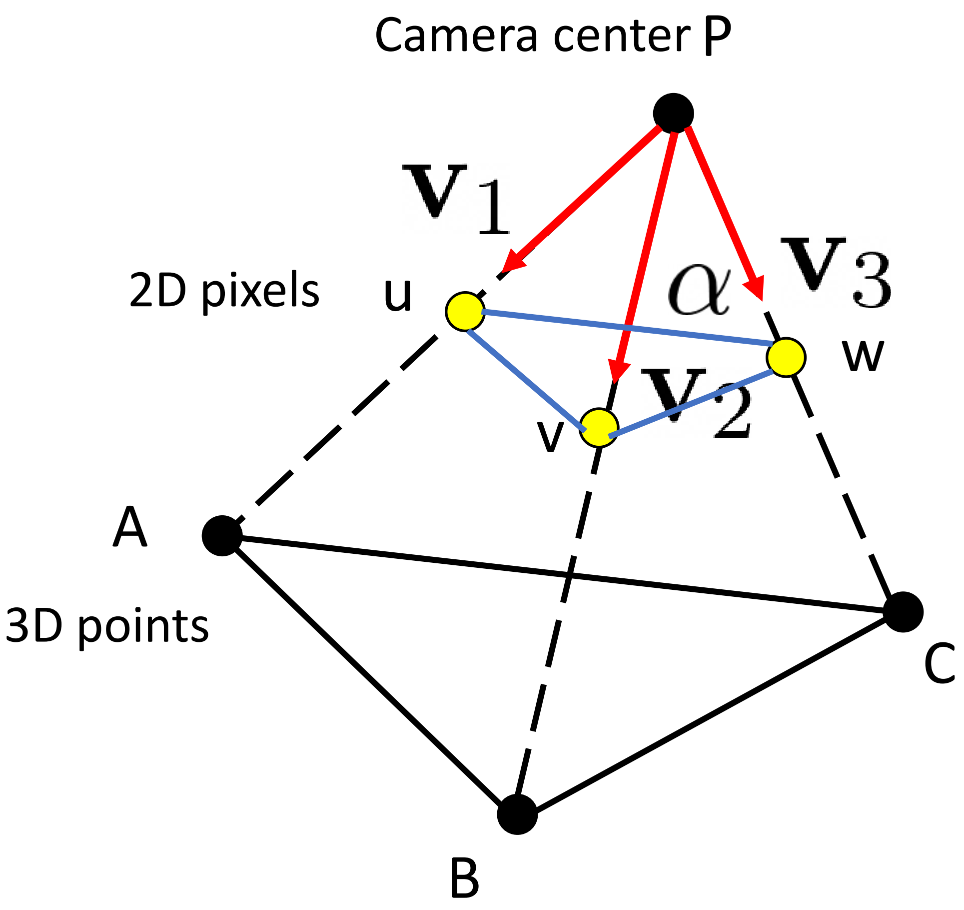
The PnP Problem

- Many different algorithms to solve the PnP problem

- General idea
  - Retrieve the coordinates of the 3D points in the camera coordinate system $\mathbf{p}_i^c$
  - Compute rotation and translation that align the world coordinates and the camera coordinates

$$\mathbf{p}_i^w \xrightarrow{R, T} \mathbf{p}_i^c$$

P3P



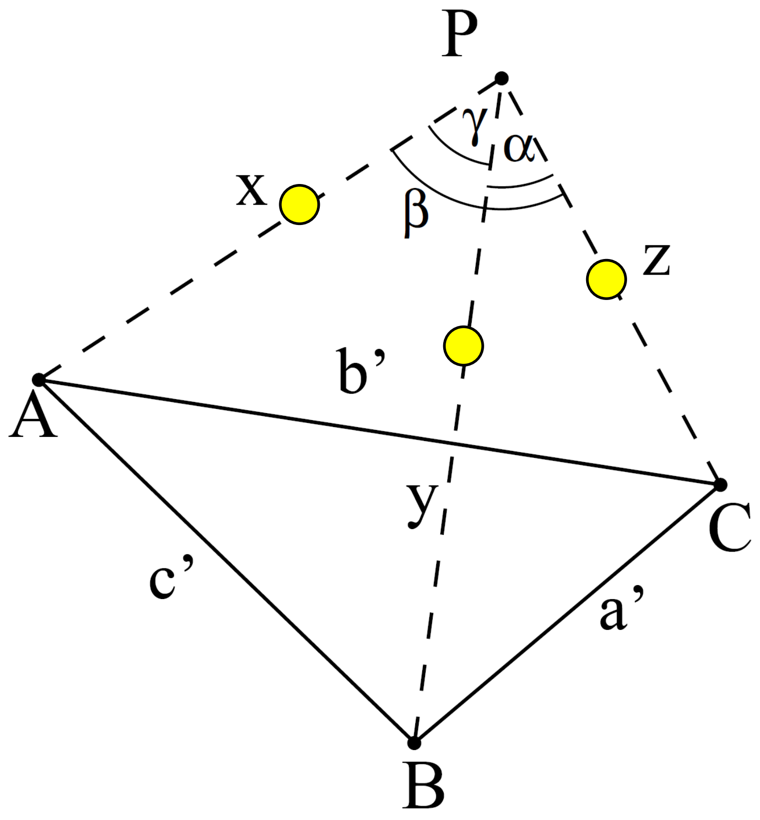
$$\mathbf{v}_1 = K^{-1}\mathbf{u}$$

$$\mathbf{v}_2 = K^{-1}\mathbf{v}$$

$$\mathbf{v}_3 = K^{-1}\mathbf{w}$$

$$\mathbf{v}_2 \cdot \mathbf{v}_3 = \|\mathbf{v}_2\| \|\mathbf{v}_3\| \cos\alpha$$

P3P



$$X = |PA| \quad Y = |PB| \quad Z = |PC|$$

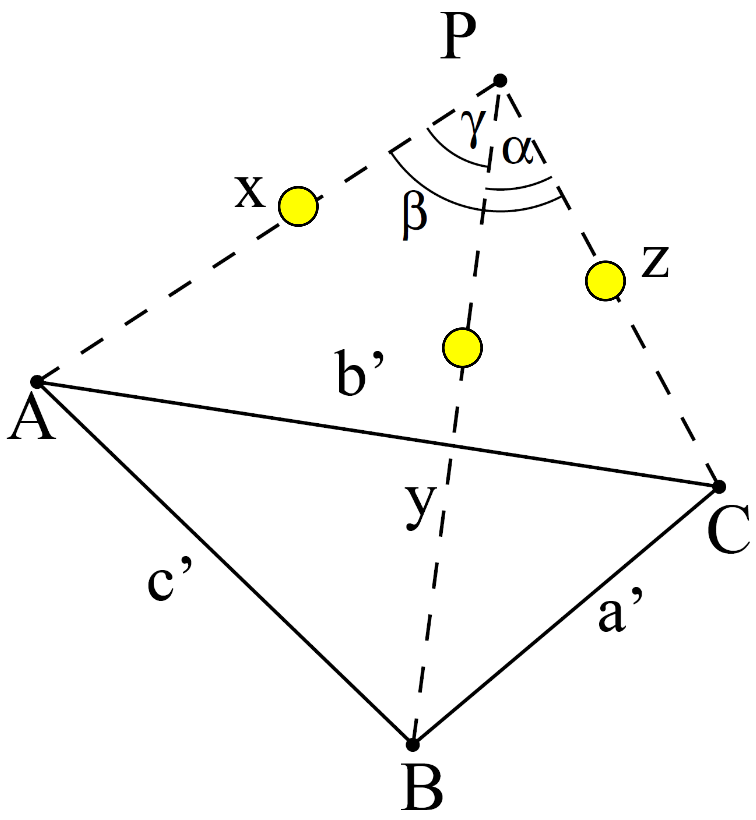
Depths of the 3 pixels

X, Y, Z are the unknowns

law of cosines
$$\begin{cases} Y^2 + Z^2 - YZp - a'^2 = 0 \\ Z^2 + X^2 - XZq - b'^2 = 0 \\ X^2 + Y^2 - XYr - c'^2 = 0. \end{cases}$$

$$p = 2\cos\alpha \qquad a' = |BC|$$
$$q = 2\cos\beta \qquad b' = |AC|$$
$$r = 2\cos\gamma \qquad c' = |AB|$$

P3P



- Find the solutions for X, Y, Z (depth of the 3 pixels)
- Obtain the coordinates of A, B, C in camera frame
- Compute R and T using the coordinates of A, B, C in camera frame and in world frame

Rotation and Translation from Two Point Sets

$$\mathbf{p}_i^w \xrightarrow{R,T} \mathbf{p}_i^c$$

Closed-form solution

K.S. Arun, T.S. Huang, and S.D. Blostein. Least-Squares Fitting of Two 3-D Points Sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):698–700, 1987.

$$\Sigma^2 = \sum_{i=1}^{N} \left\| p_i' - (Rp_i + T) \right\|^2.$$

Or https://cs.gmu.edu/~kosecka/cs685/cs685-icp.pdf

EPnP

- EPnP: uses 4 control points $\mathbf{c}_j, \quad j = 1, \ldots, 4$

3D coordinates in the world frame $\mathbf{p}_i^w = \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j^w$ Known

Weights $\sum_{j=1}^{4} \alpha_{ij} = 1$ Known

3D coordinates in the camera frame $\mathbf{p}_i^c = \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j^c$ Unknown

EP*n*P: An Accurate O(n) Solution to the P*n*P Problem. Lepetit et al., IJCV'09.

EPnP

- Projection of the points in the camera frame

$$\forall i \ , \ w_i \begin{bmatrix} \mathbf{u}_i \\ 1 \end{bmatrix} = K\mathbf{p}_i^c = K \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j^c$$

$$\forall i \ , \ w_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_c \\ 0 & f_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \sum_{j=1}^{4} \alpha_{ij} \begin{bmatrix} x_j^c \\ y_j^c \\ z_j^c \end{bmatrix}$$

Unknown $\{(x_j^c, y_j^c, z_j^c)\}_{j=1,\ldots,4}$ $\{w_i\}_{i=1,\ldots,n}$ $w_i = \sum_{j=1}^{4} \alpha_{ij} z_j^c$

EP*n*P: An Accurate O(n) Solution to the P*n*P Problem. Lepetit et al., IJCV'09.

EPnP

$$\forall i \ , \ w_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_c \\ 0 & f_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \sum_{j=1}^{4} \alpha_{ij} \begin{bmatrix} x_j^c \\ y_j^c \\ z_j^c \end{bmatrix}$$

$$w_i = \sum_{j=1}^{4} \alpha_{ij} z_j^c$$

$$\sum_{j=1}^{4} \alpha_{ij} f_u x_j^c + \alpha_{ij}(u_c - u_i) z_j^c = 0$$

$$\sum_{j=1}^{4} \alpha_{ij} f_v y_j^c + \alpha_{ij}(v_c - v_i) z_j^c = 0$$

Unknown $\left\{ (x_j^c, y_j^c, z_j^c) \right\}_{j=1,\dots,4}$

$$\mathbf{M}\mathbf{x} = \mathbf{0}$$

$$\mathbf{x} = \left[\mathbf{c}_1^{c\top}, \mathbf{c}_2^{c\top}, \mathbf{c}_3^{c\top}, \mathbf{c}_4^{c\top} \right]^\top \ 12 \times 1$$

$\mathbf{M}$ is a $2n \times 12$ matrix

EP*n*P: An Accurate O(n) Solution to the P*n*P Problem. Lepetit et al., IJCV'09.

EPnP

- Solve $\mathbf{M}\mathbf{x} = \mathbf{0}$ to obtain $\mathbf{x} = [\mathbf{c}_1^{c\top}, \mathbf{c}_2^{c\top}, \mathbf{c}_3^{c\top}, \mathbf{c}_4^{c\top}]^\top$ See. Lepetit et al., IJCV'09
- Compute 3D coordinates in camera frame $\mathbf{p}_i^c = \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j^c$
- We know the 3D coordinates in world frame $\mathbf{p}_i^w = \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j^w$
- Compute R and T using the two sets of 3D coordinates

$$\mathbf{p}_i^w \xrightarrow{R, T} \mathbf{p}_i^c$$

EP*n*P: An Accurate O(n) Solution to the P*n*P Problem. Lepetit et al., IJCV'09.

PnP in practice

- SolvePnPMethod in OpenCV

◆ SolvePnPMethod

enum **cv::SolvePnPMethod**

```
#include <opencv2/calib3d.hpp>
```

Enumerator	
SOLVEPNP_ITERATIVE Python: cv.SOLVEPNP_ITERATIVE	
SOLVEPNP_EPNP Python: cv.SOLVEPNP_EPNP	EPnP: Efficient Perspective-n-Point Camera Pose Estimation [125].
SOLVEPNP_P3P Python: cv.SOLVEPNP_P3P	Complete Solution Classification for the Perspective-Three-Point Problem [80].
SOLVEPNP_DLS Python: cv.SOLVEPNP_DLS	**Broken implementation. Using this flag will fallback to EPnP.** A Direct Least-Squares (DLS) Method for PnP [101]
SOLVEPNP_UPNP Python: cv.SOLVEPNP_UPNP	**Broken implementation. Using this flag will fallback to EPnP.** Exhaustive Linearization for Robust Camera Pose and Focal Length Estimation [169]
SOLVEPNP_AP3P Python: cv.SOLVEPNP_AP3P	An Efficient Algebraic Solution to the Perspective-Three-Point Problem [114].
SOLVEPNP_IPPE Python: cv.SOLVEPNP_IPPE	Infinitesimal Plane-Based Pose Estimation [46] Object points must be coplanar.
SOLVEPNP_IPPE_SQUARE Python: cv.SOLVEPNP_IPPE_SQUARE	Infinitesimal Plane-Based Pose Estimation [46] This is a special case suitable for marker pose estimation. 4 coplanar object points must be defined in the following order: • point 0: [-squareLength / 2, squareLength / 2, 0] • point 1: [squareLength / 2, squareLength / 2, 0] • point 2: [squareLength / 2, -squareLength / 2, 0] • point 3: [-squareLength / 2, -squareLength / 2, 0]
SOLVEPNP_SQPNP Python: cv.SOLVEPNP_SQPNP	SQPnP: A Consistently Fast and Globally OptimalSolution to the Perspective-n-Point Problem [208].

QR Code Pose Tracking Example

https://levelup.gitconnected.com/qr-code-scanner-in-kotlin-e15dd9bfbb1f

Further Reading

- Stanford CS231A: Computer Vision, From 3D Reconstruction to Recognition, Lecture 3 https://web.stanford.edu/class/cs231a/syllabus.html
- A Flexible New Technique for Camera Calibration. Zhengyou Zhang, TPAMI. 2000.
- EP*n*P: An Accurate O(n) Solution to the P*n*P Problem. Lepetit et al., IJCV'09.