Augmented Reality

3A SN M + 3EA IATI

Simone Gasparini
simone.gasparini@toulouse-inp.fr

Today Plan

- The registration problem
- Camera visual tracking for AR
  - Background – camera model and calibration
  - Fiducials based tracking
  - Marker based tracking

- Next class:
  - Visual SLAM

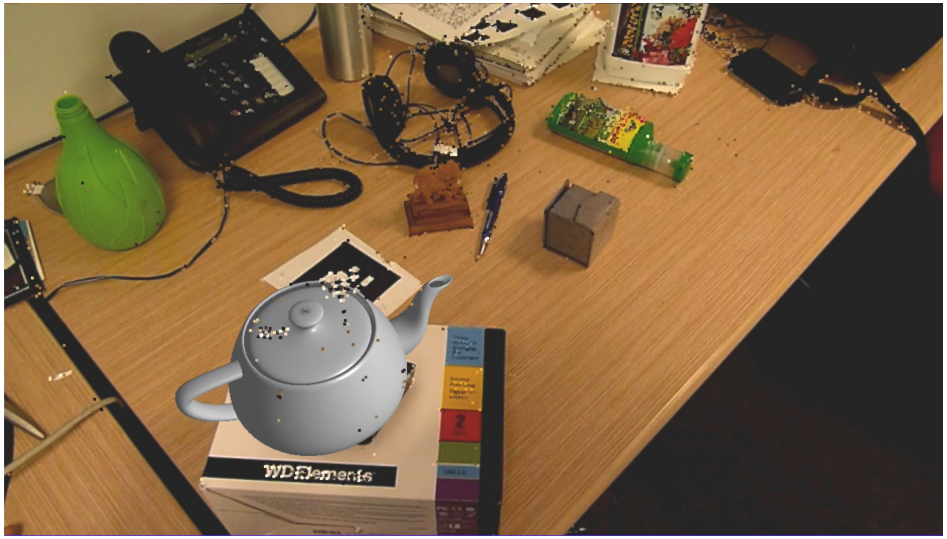
Previously…

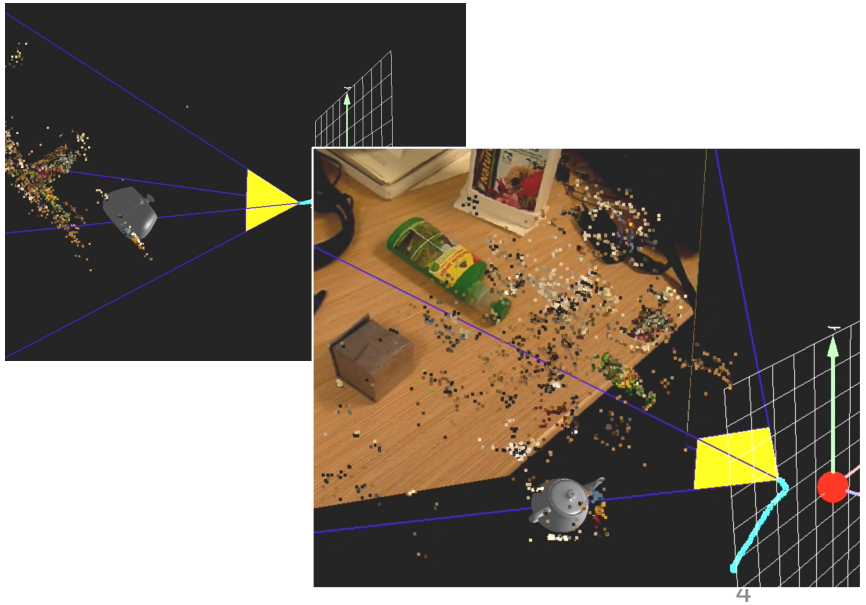
- Augmented reality
  - Inserting virtual content in the real world
  - Real-time
  - Registered in 3D
  - Eventually Interactive

The registration problem

- The problem: inserting a 3D virtual object inside an image
- The object has to look “real”
  - Appearance
  - It must look **integrated** in the scene
    - Robust to changes of the point of view
    - Eventually, deal with occlusions, light sources etc

https://youtu.be/-NS7vzk0t0I





Project ROM @IRIT eg. Vortex-N7


FLEURS

The registration problem

- The main idea:
  - Project the 3D model of the object through the camera and render it

We need:

- The 3D model of the object ✔
  - 3D mesh + textures + properties
- The camera model ✔
  - The intrinsic parameters
  - The optical distortion parameters
  - [advanced] other settings (focus, depth of focus etc)
- The position
  - Of the object in the scene
  - Of the camera wrt the scene

The registration problem

- Known world assumption
  - A 3D model of the scene is known (eg. Depth sensors kinect-like)
  - The object model is placed in the scene
  - Only camera position and orientation to track
  - Ideal case for dealing with occlusions

KinectFusion

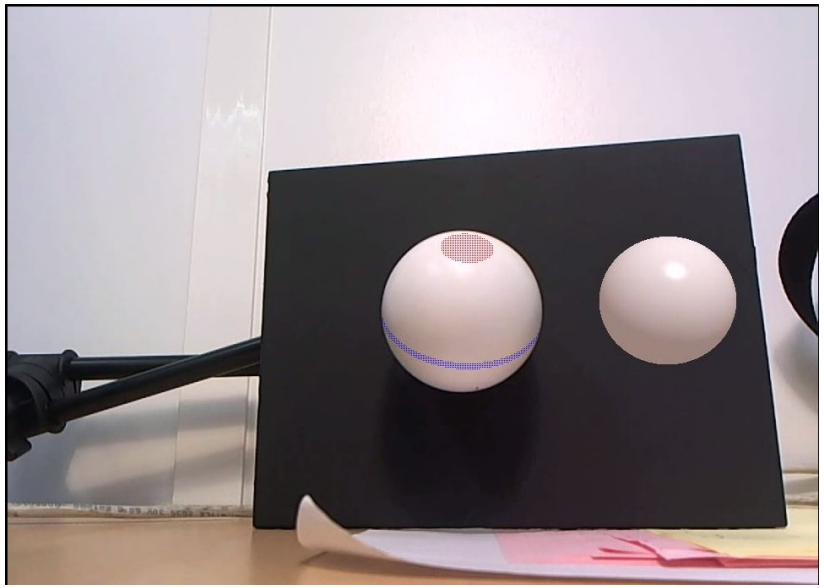
Shahram Izadi, Richard A. Newcombe, David Kim, Otmar Hilliges, David Molyneaux, Steve Hodges, Pushmeet Kohli, Jamie Shotton, Andrew J. Davison, and Andrew Fitzgibbon. 2011. KinectFusion: real-time dynamic 3D surface reconstruction and interaction. In *ACM SIGGRAPH 2011 Talks* (SIGGRAPH '11).

Real-time with 8 cpu and 2 gpu…

http://youtu.be/q8jfgTilFmo?t=2m1s

The registration problem

- Going beyond: Light estimation

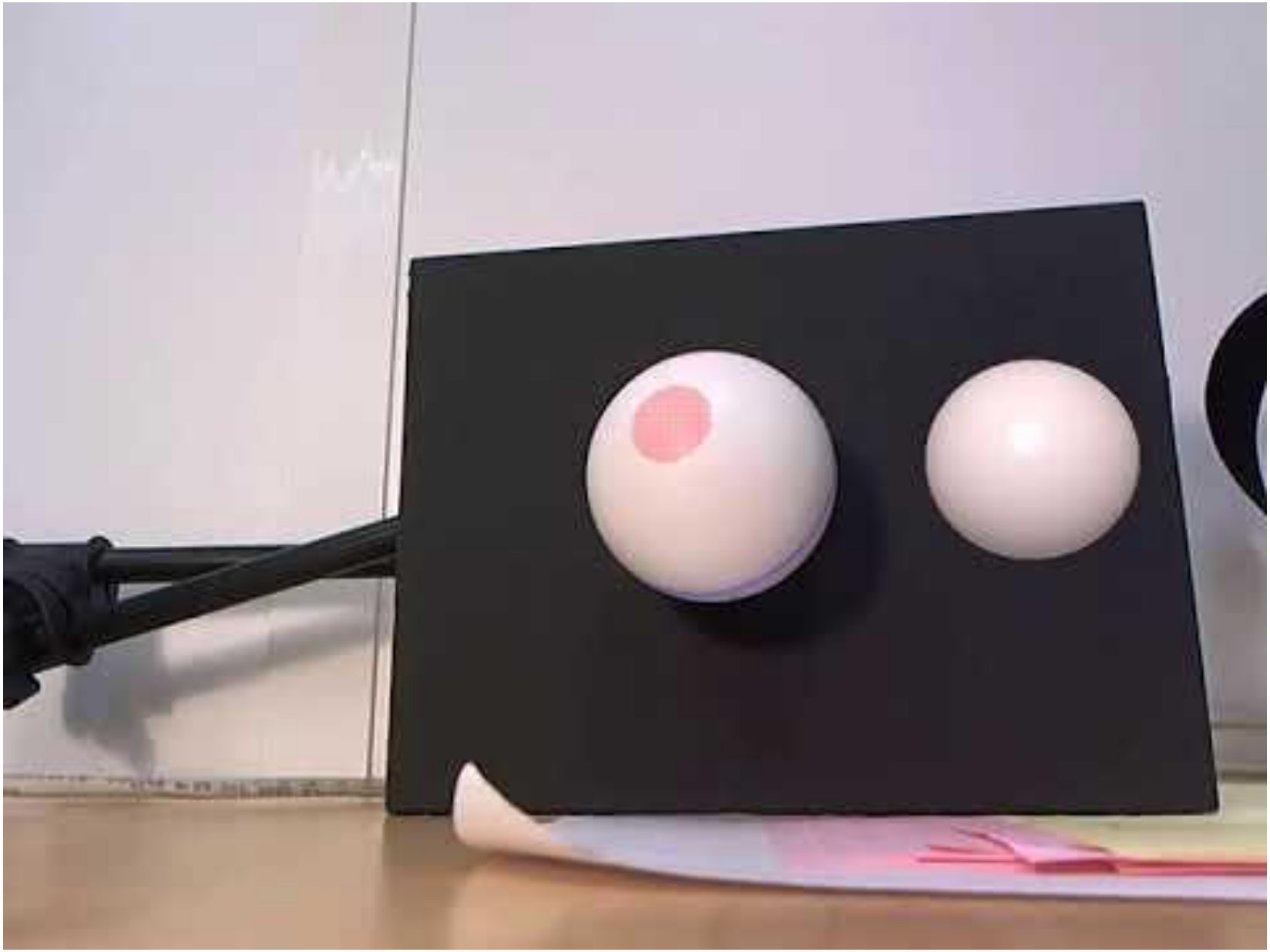


https://youtu.be/v0HsU8AluAQ

Light direction estimation @IRIT (J.D. Durou)

https://www.youtube.com/watch?v=PyP9AwECK2g

Instant Reality rendering coupled with light estimation




instantreality

The registration problem

- Going even further beyond: interaction with real objects
  - Semantics!!

http://youtu.be/XZZivNnZBpk?t=43m54s

From the talk given by David Forsyth : Understanding Pictures of Rooms and Inserting Objects into them

The registration problem

- Or maybe when can simply put all together like this: ☺️

Aug(De)Mented Reality https://youtu.be/gpum4nK2wOM

Trackers

- AR system must
  - Sense the environment
  - Track the position of user (6DOF)
- Many choices:
  - Mechanical, ultrasonic, magnetic
  - GPS
  - Radio
  - Inertial
  - Optical
  - Hybrid
- Depends on applications
  - Indoor Vs Outdoor
  - Level of precision: depends on the distance
- **No best solution!**

Trackers – **Optical trackers**

- Use the camera feed to track the movement
- Track:
  - Artificial markers
    - Easy to detect markers in real-time
    - Need to set up the environment
  - Natural features
    - not easy
    - More freedom but need textured environment
- Good accuracy for slower motions
  - Accuracy affected by
    - Blur (motion, focus)
    - Light changes
    - Occlusions
    - Textureless objects/environment

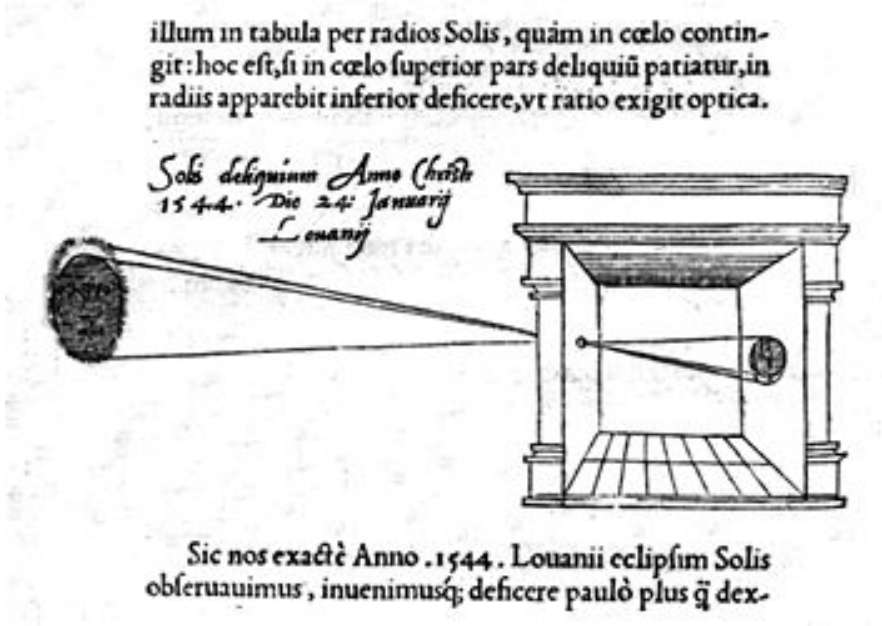
The registration problem

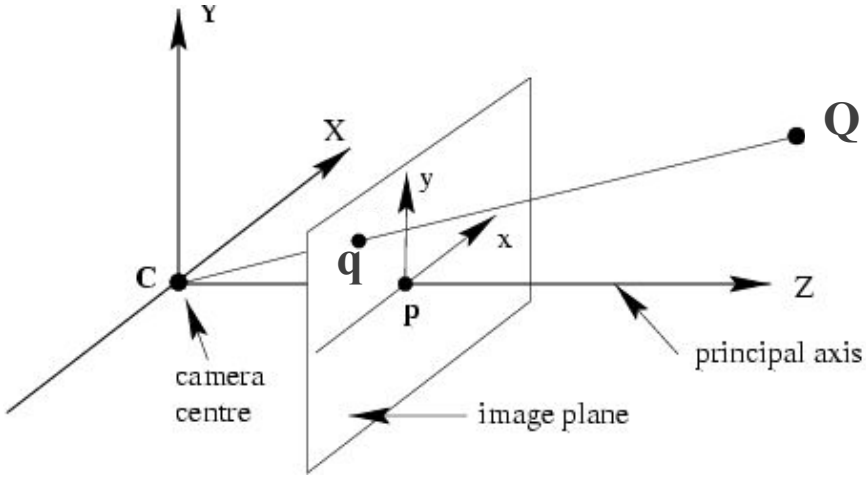
- Main issues to solve is the camera pose estimation
- Camera tracking (*suivi de caméra*)
  - Estimate the camera position and orientation
- Different technologies
  - Type of tracker
- Different algorithms according to the knowledge about the scene
  (What do we know?)
  - Some 3D references in the scene (fiducials)
  - The 3D model of an object to track (model-based tracking)
  - Some 2D reference in the scene (markers)
  - Nothing? Use the natural features (SLAM)

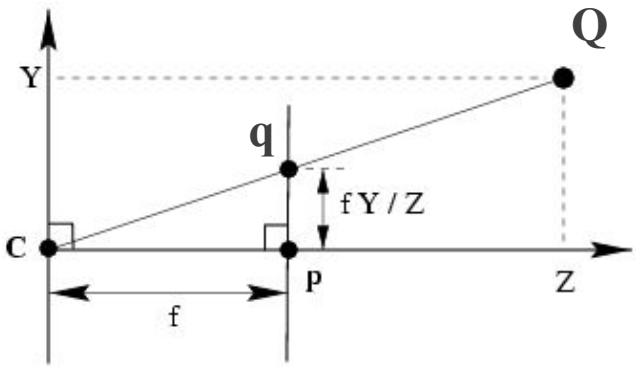
Today Topics

- The registration problem
- Optical visual tracking for AR
  - Background – camera model and calibration
  - Pose estimation using fiducials
    - Model Based tracking
  - Pose estimation using markers

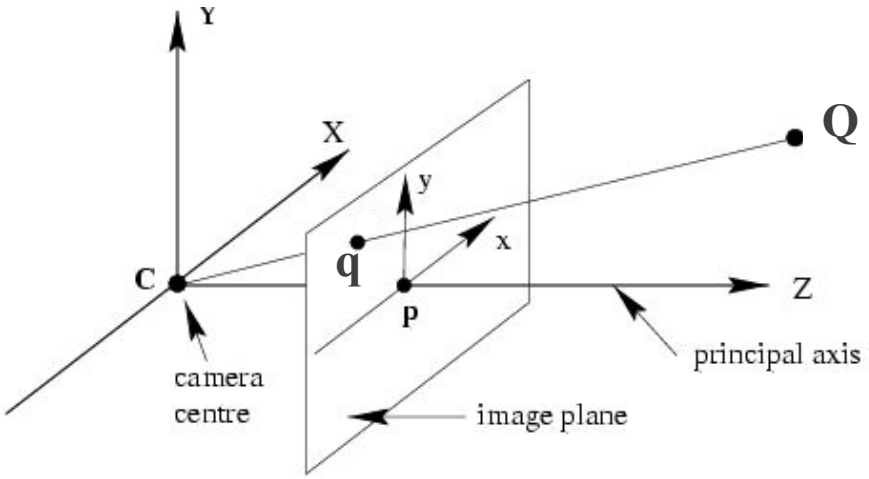
Pinhole camera

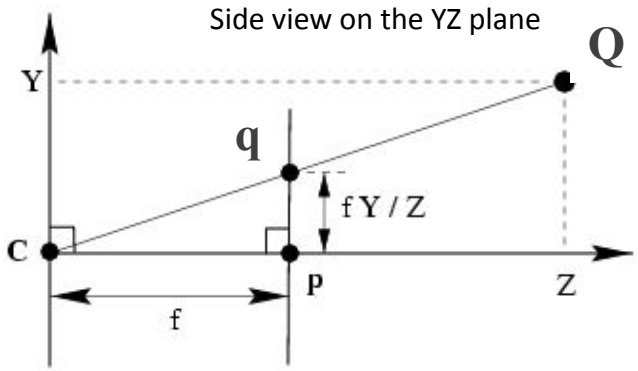




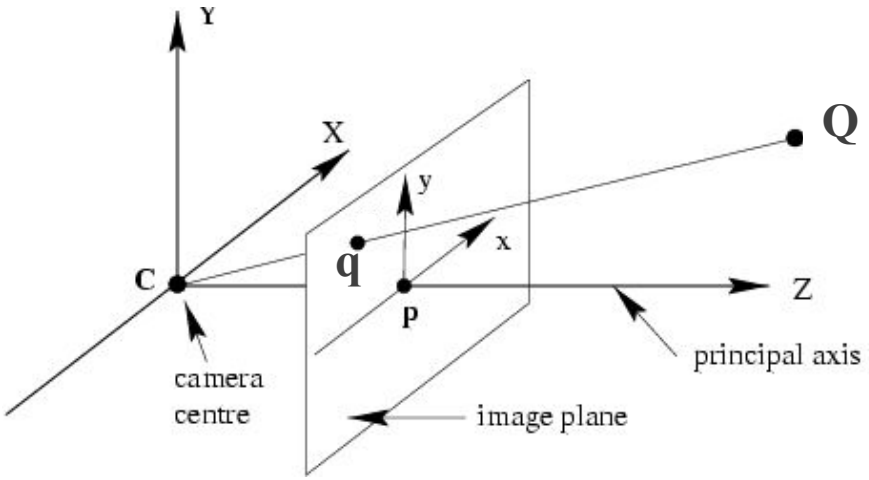


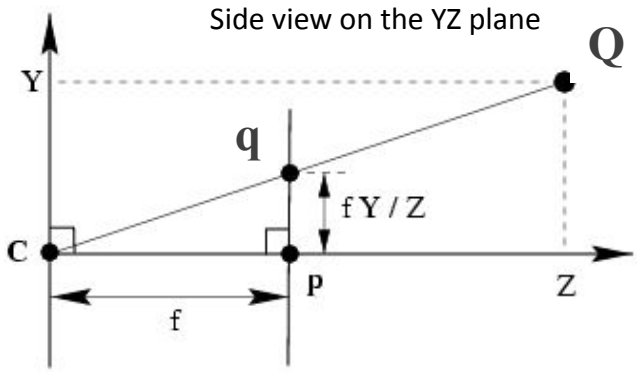
Pinhole camera





Pinhole camera



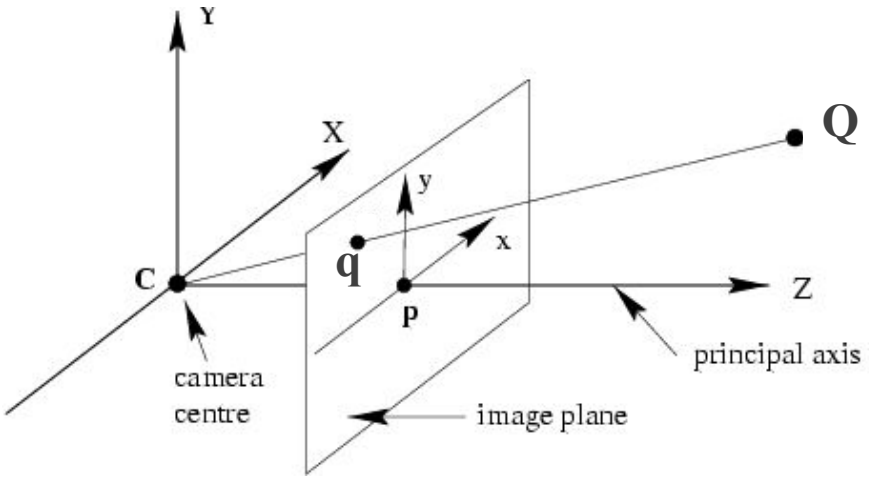


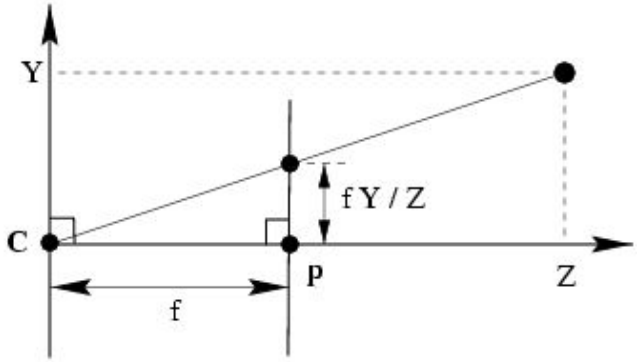
$$(Q_x, Q_y, Q_z)^T \mapsto (fQ_x / Q_z, fQ_y / Q_z)^T$$

Using homogeneous coordinates

$$\begin{pmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fQ_x \\ fQ_y \\ Q_z \end{pmatrix}$$

Pinhole camera



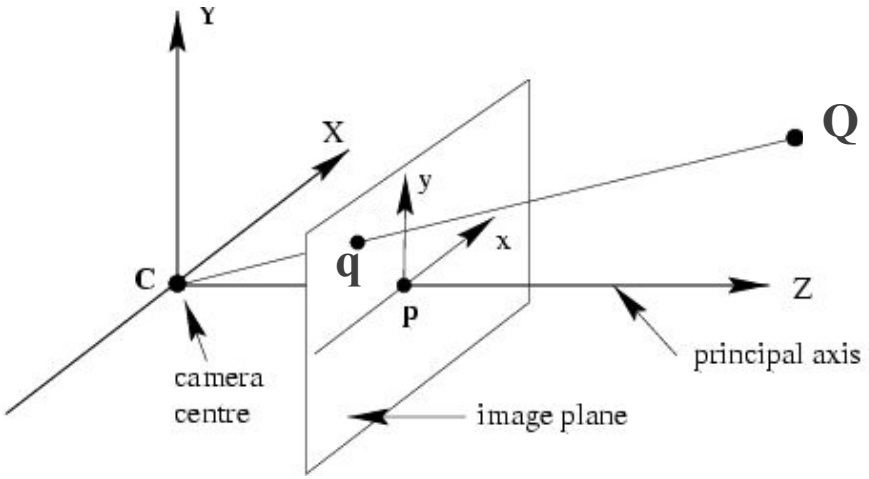


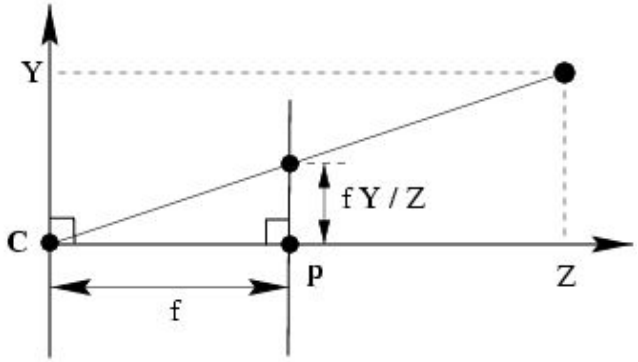
$$(Q_x, Q_y, Q_z)^T \mapsto (fQ_x / Q_z, fQ_y / Q_z)^T$$

Using homogeneous coordinates

$$\begin{pmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fQ_x \\ fQ_y \\ Q_z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{pmatrix}$$

Pinhole camera



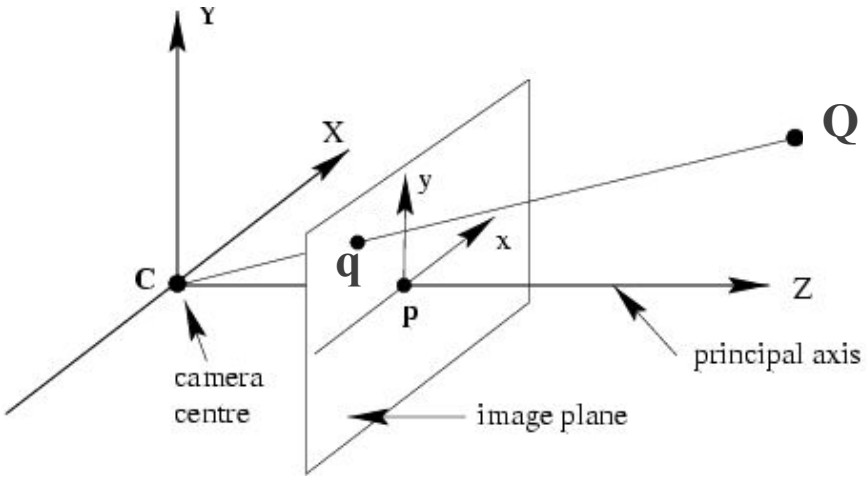


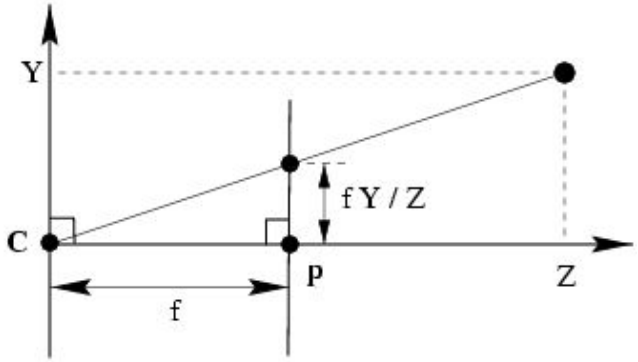
$$(Q_x, Q_y, Q_z)^T \mapsto (fQ_x / Q_z, fQ_y / Q_z)^T$$

Using homogeneous coordinates

$$\begin{pmatrix} fQ_x \\ fQ_y \\ Q_z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{pmatrix}$$

Pinhole camera

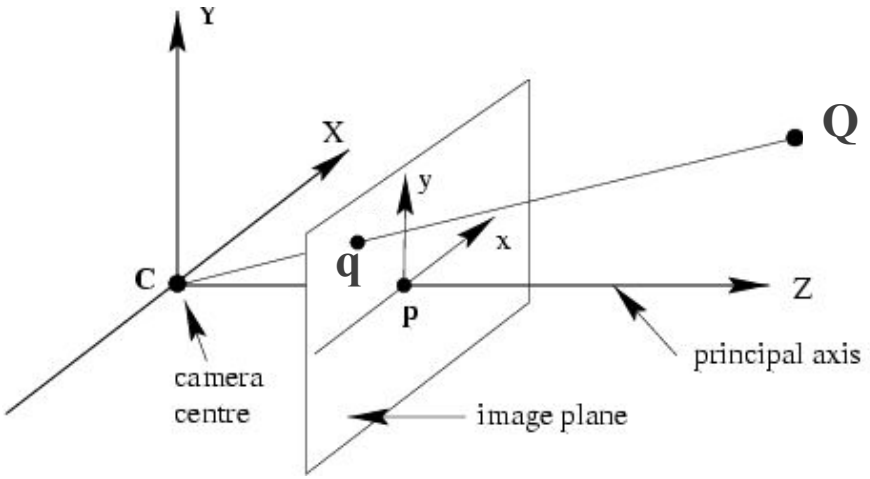


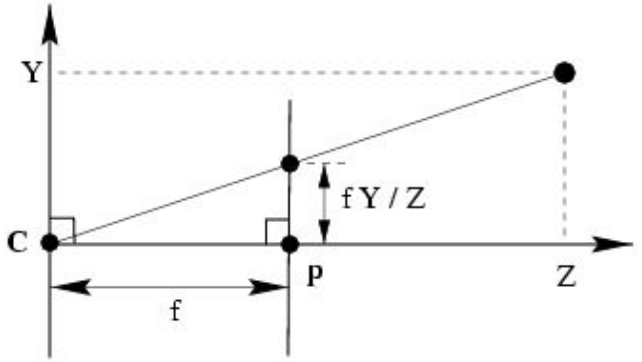


$$(Q_x, Q_y, Q_z)^T \mapsto (fQ_x / Q_z, fQ_y / Q_z)^T$$

$$\begin{pmatrix} fQ_x/Q_z \\ fQ_y/Q_z \\ 1 \end{pmatrix} = \begin{pmatrix} q_x \\ q_y \\ 1 \end{pmatrix} = \lambda \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{pmatrix} \quad with\ \lambda = \frac{1}{Q_z}$$

Pinhole camera



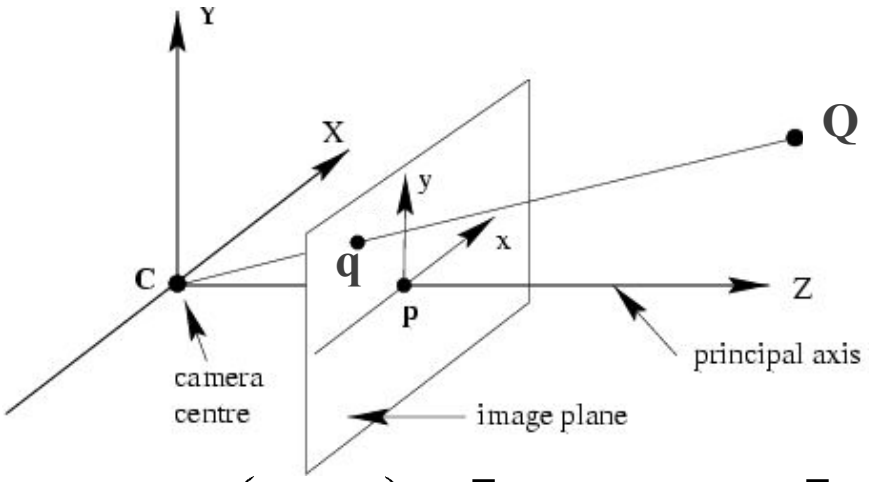


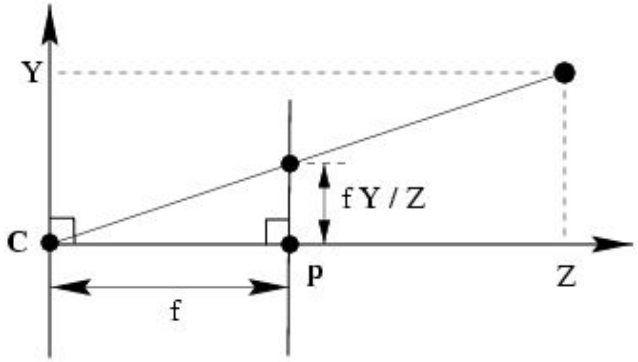
$$(Q_x, Q_y, Q_z)^T \mapsto (fQ_x / Q_z, fQ_y / Q_z)^T$$

$$\begin{pmatrix} q_x \\ q_y \\ 1 \end{pmatrix} \sim \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{pmatrix}$$

with " $\sim$ " up to a scale factor $\lambda = \dfrac{1}{Q_z}$

Pinhole camera



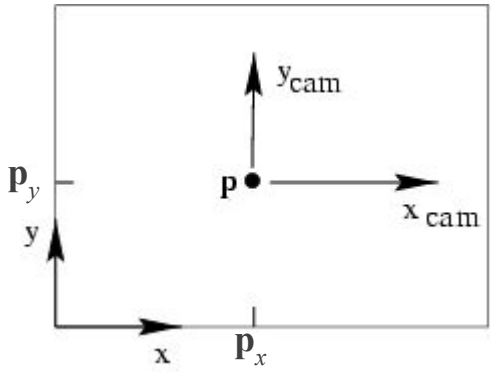


$$\begin{pmatrix} q_x \\ q_y \\ 1 \end{pmatrix} \sim \begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{pmatrix}$$

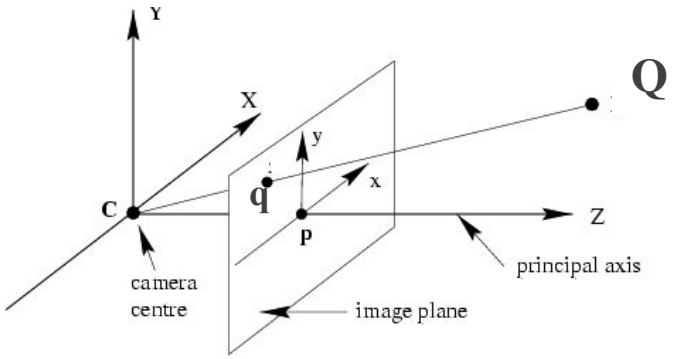
$$\mathbf{q} \sim \underbrace{\mathbf{K}_{3\times3}\left[\mathbf{I}_{3\times3} \mid \mathbf{0}_{3\times1}\right]}_{\mathbf{P}}\mathbf{Q} = \mathbf{P}\mathbf{Q}$$

P projection matrix

Principal point offset



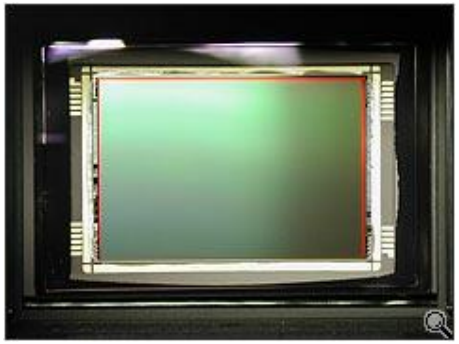
p principal point



$$(Q_x, Q_y, Q_z)^T \mapsto (fQ_x / Q_z + p_x, Q_y / Q_z + p_y)^T$$

$$\begin{pmatrix} q_y \\ q_y \\ 1 \end{pmatrix} \sim \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{pmatrix}$$

CCD cameras – non square pixels



$$\mathbf{K} = \begin{bmatrix} m_x & & \\ & m_y & \\ & & 1 \end{bmatrix} \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}$$

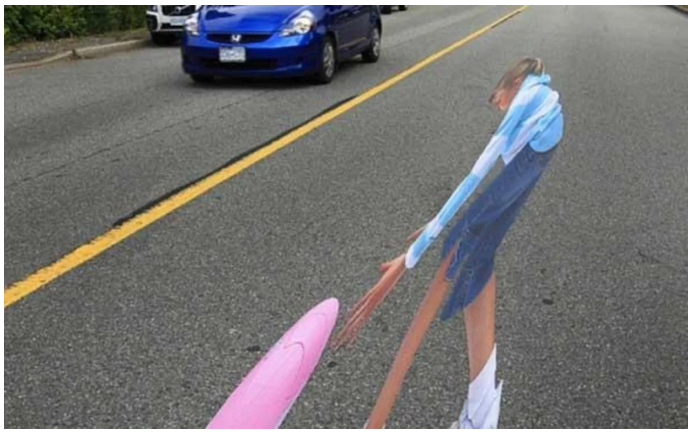
$$\mathbf{K} = \begin{bmatrix} \alpha_x & & p_x \\ & \alpha_y & p_y \\ & & 1 \end{bmatrix}$$

Internal or **Intrinsics** parameters of the camera

Scale factor ambiguity

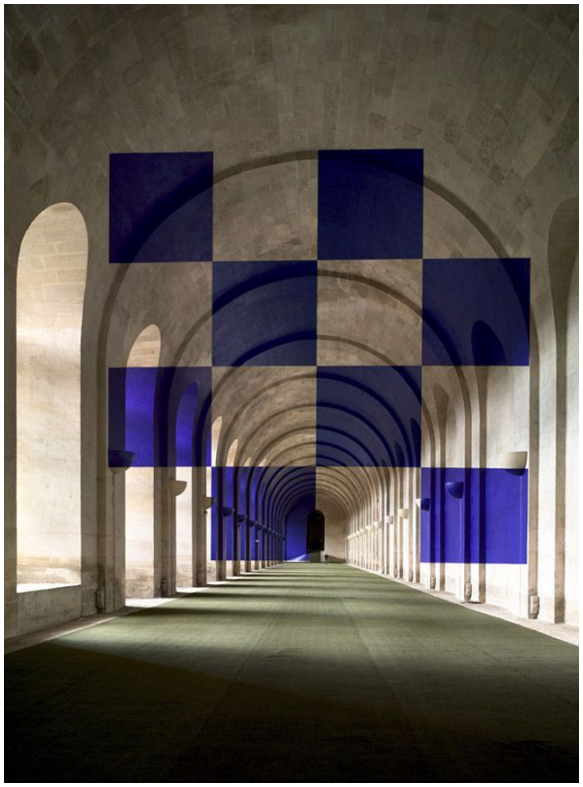
Perspective ambiguity

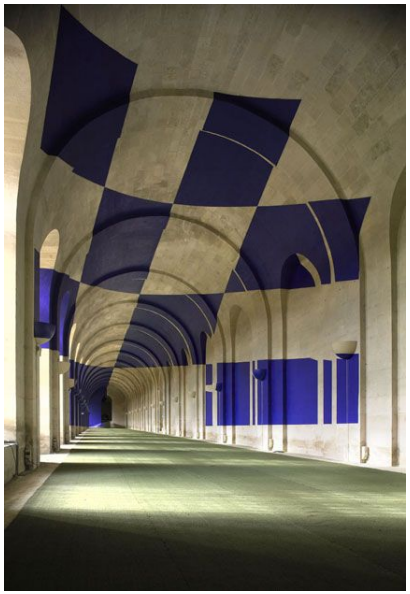
Anamorphosis



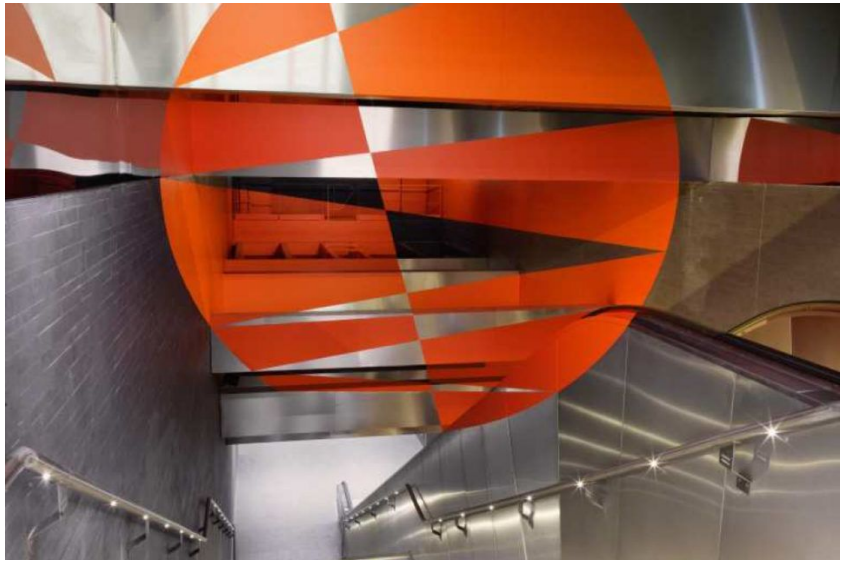
[This 3D illusion of a little girl in the road gives drivers serious pause - Vancouver Is Awesome]

Perspective ambiguity



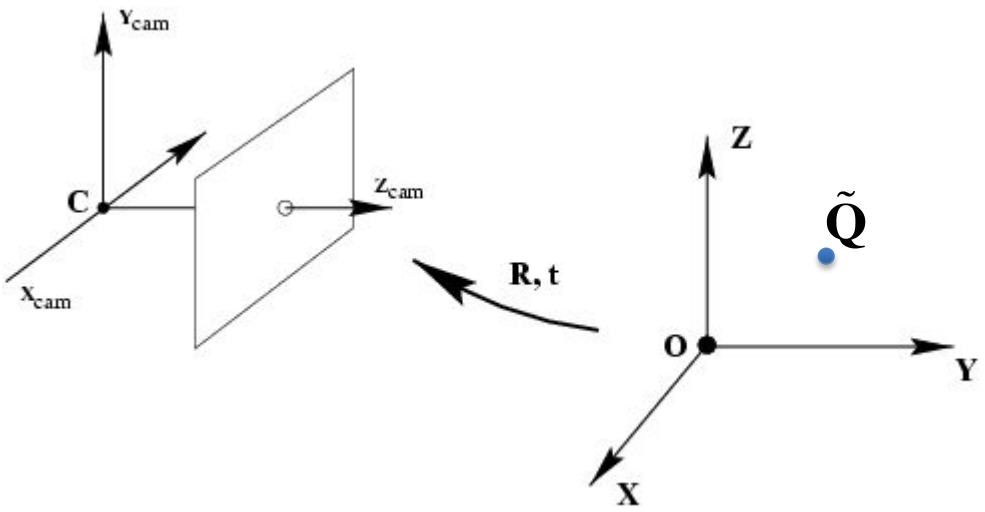


Felice Varini - Huit Carrés,
Orangerie du château de Versailles



Felice Varini - Zigzag dans le disque,
Station Jean Jaurès, Toulouse

Camera rotation and translation

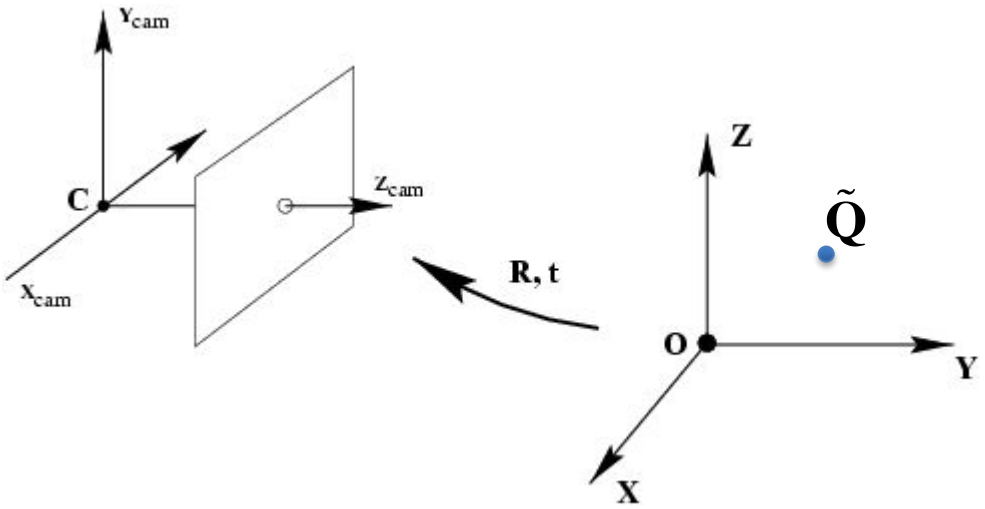


$\tilde{\mathbf{Q}}_{\text{cam}}$ A 3d point in camera ref system

$\tilde{\mathbf{Q}}$ A 3d point in world ref system

$\tilde{\mathbf{C}}$ Camera center in world ref system

Camera rotation and translation



$\tilde{\mathbf{Q}}_{\text{cam}}$ A 3d point in camera ref system

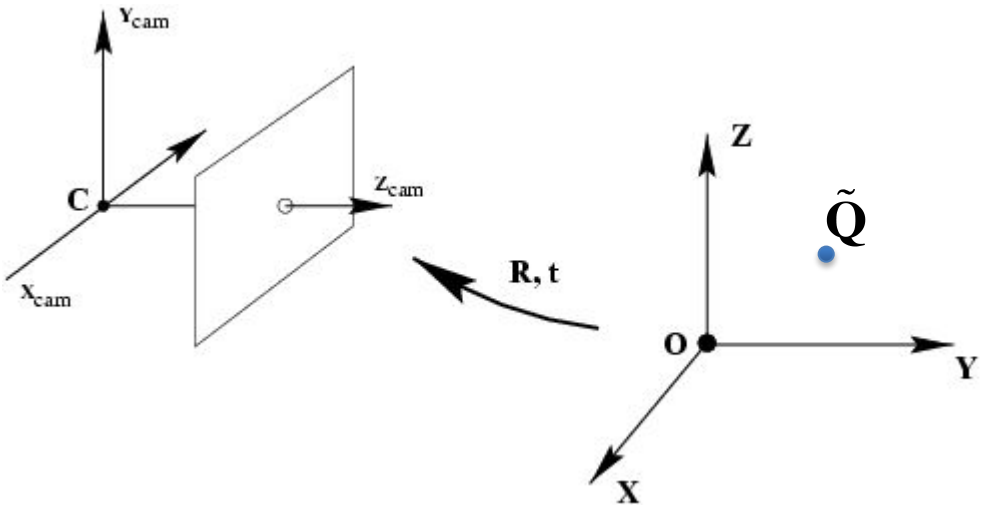
$\tilde{\mathbf{Q}}$ A 3d point in world ref system

$\tilde{\mathbf{C}}$ Camera center in world ref system

$$\tilde{\mathbf{Q}}_{\text{cam}} = \mathbf{R}\left(\tilde{\mathbf{Q}} - \tilde{\mathbf{C}}\right)$$

$$\tilde{\mathbf{Q}}_{\text{cam}} = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\tilde{\mathbf{C}} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} \tilde{Q}_x \\ \tilde{Q}_y \\ \tilde{Q}_z \\ 1 \end{pmatrix} = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\tilde{\mathbf{C}} \\ 0 & 1 \end{bmatrix} \tilde{\mathbf{Q}}$$

Camera rotation and translation



$\tilde{\mathbf{Q}}_{\text{cam}}$ A 3d point in camera ref system

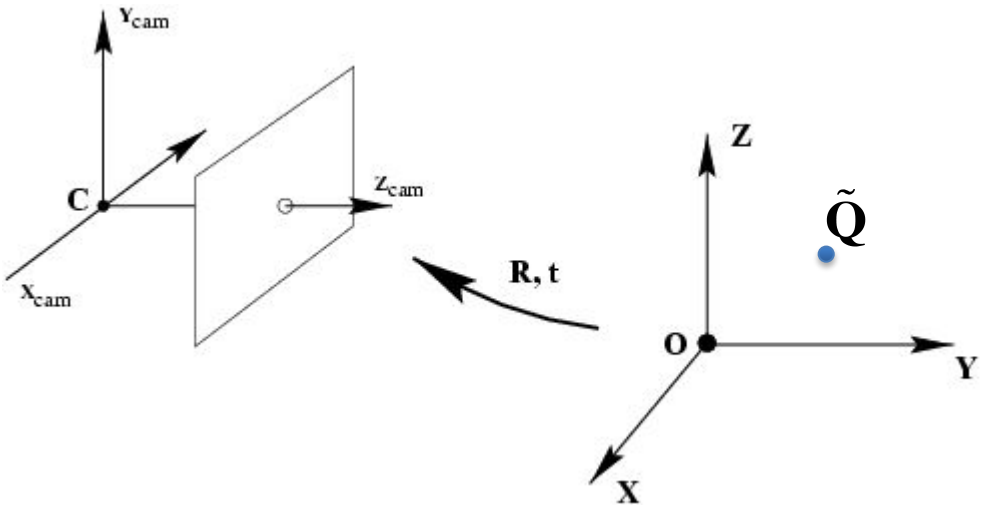
$\tilde{\mathbf{Q}}$ A 3d point in world ref system

$\tilde{\mathbf{C}}$ Camera center in world ref system

$$\tilde{\mathbf{Q}}_{\text{cam}} = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\tilde{\mathbf{C}} \\ 0 & 1 \end{bmatrix} \tilde{\mathbf{Q}}$$

$$\mathbf{q} \sim \mathbf{P}\tilde{\mathbf{Q}}_{\text{cam}} = \mathbf{K}[\mathbf{I}\,|\,\mathbf{0}]\tilde{\mathbf{Q}}_{\text{cam}} = \mathbf{K}[\mathbf{I}\,|\,\mathbf{0}] \begin{bmatrix} \mathbf{R} & -\mathbf{R}\tilde{\mathbf{C}} \\ 0 & 1 \end{bmatrix} \tilde{\mathbf{Q}}$$

Camera rotation and translation



$\tilde{\mathbf{Q}}_{\text{cam}}$ A 3d point in camera ref system

$\tilde{\mathbf{Q}}$ A 3d point in world ref system

$\tilde{\mathbf{C}}$ Camera center in world ref system

$$\mathbf{q} \sim \mathbf{P}\tilde{\mathbf{Q}}_{\text{cam}} = \mathbf{K}\left[\mathbf{I}\,|\,\mathbf{0}\right]\tilde{\mathbf{Q}}_{\text{cam}} = \mathbf{K}\left[\mathbf{I}\,|\,\mathbf{0}\right]\begin{bmatrix}\mathbf{R} & -\mathbf{R}\tilde{\mathbf{C}} \\ 0 & 1\end{bmatrix}\tilde{\mathbf{Q}}$$

$$\mathbf{q} \sim \underbrace{\mathbf{K}\left[\mathbf{R}\,|\,\mathbf{t}\right]}_{\mathbf{P}_{3\times 4}}\mathbf{Q} \qquad \mathbf{t} = -\mathbf{R}\tilde{\mathbf{C}}$$

Action of a projective camera on points

Forward projection

- In camera coordinates:

$$q \sim \mathbf{K}[\mathbf{I}\,|\,\mathbf{0}]\mathbf{Q} = \mathbf{K}\tilde{\mathbf{Q}} \quad \text{with} \quad \mathbf{Q} = \begin{bmatrix} \tilde{\mathbf{Q}} \\ 1 \end{bmatrix}$$

$$\mathbf{K}^{-1}q \sim \tilde{\mathbf{Q}} \qquad \mathbf{K}^{-1}q$$ Is the direction of the 3D projection ray of **Q**

- Point in space mapped according to

$$x = \mathbf{P}X$$

- Points at infinity

$$x = \mathbf{P}D = \mathbf{K}[\mathbf{R}\,|\,\mathbf{t}]\begin{pmatrix} \mathbf{d} \\ 0 \end{pmatrix} = \mathbf{KRd}$$

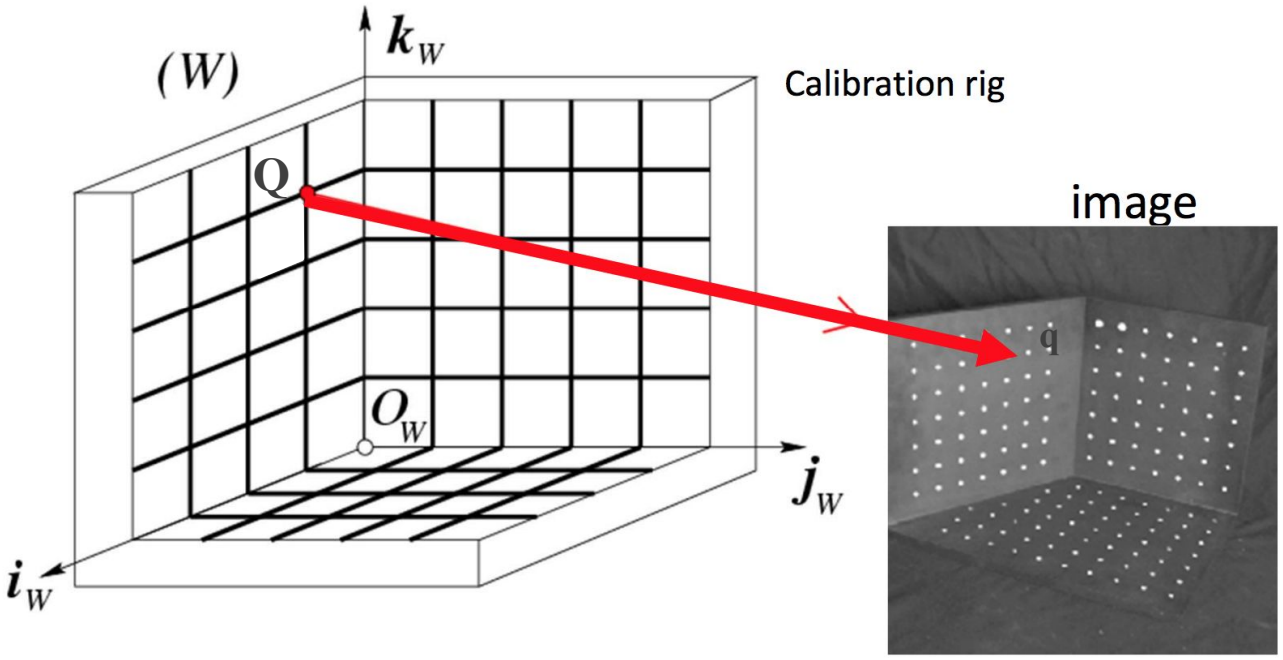
Camera calibration

- How can we estimate the internal parameters of the camera?
- From the projection model:
  - **q** is known as we can measure points on the image
  - **Q** ? Can we know **Q**?
  - **K** internal parameters (intrinsics) --> to be estimated
  - **R** and t external parameters (extrinsics) --> to be estimated

$$\mathbf{q} \sim \underbrace{\mathbf{K}\left[\mathbf{R} \mid \mathbf{t}\right]}_{\mathbf{P}_{3\times 4}}\mathbf{Q}$$

Camera calibration

- We can devise a calibration device for which the points **Q** are known with good enough accuracy
- For example:



Camera calibration

- How can we estimate the internal parameters of the camera?
- We can have as many as we want associations **q** <-> **Q**
- Only **P** which contains **K**, **R** and t is left to be estimated
- What is the *minimum number of associations* **q** <-> **Q** necessary to estimate **P**?
- Let's develop the projection equation...

$$\mathbf{q} \sim \underbrace{\mathbf{K}\begin{bmatrix}\mathbf{R} \mid \mathbf{t}\end{bmatrix}}_{\mathbf{P}_{3\times 4}}\mathbf{Q}$$

Camera calibration

For each pair of correspondences $\mathbf{q}_i \leftrightarrow \mathbf{Q}_i$

$$\lambda_i \mathbf{q}_i = \mathbf{P}_{3\times 4}\mathbf{Q}_i = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \mathbf{p}_3^T \end{bmatrix} \mathbf{Q}_i$$

Developing and considering that $q_z = 1$ we can get rid of λ_i

$$\begin{cases} \lambda_i q_x = \mathbf{p}_1^T \mathbf{Q}_i \\ \lambda_i q_y = \mathbf{p}_2^T \mathbf{Q}_i \\ \lambda_i q_z = \mathbf{p}_3^T \mathbf{Q}_i \end{cases} \xrightarrow{q_z=1} \begin{cases} \cdots \\ \cdots \\ \lambda_i = \mathbf{p}_3^T \mathbf{Q}_i \end{cases} \rightarrow \begin{cases} q_x \mathbf{p}_3^T \mathbf{Q}_i - \mathbf{p}_1^T \mathbf{Q}_i = 0 \\ q_y \mathbf{p}_3^T \mathbf{Q}_i - \mathbf{p}_2^T \mathbf{Q}_i = 0 \\ \lambda_i = \mathbf{p}_3^T \mathbf{Q}_i \end{cases}$$

Collecting the terms in function of the unknowns $\mathbf{p}_i$

$$\begin{bmatrix} \mathbf{Q}_i^T & \mathbf{0}_{1\times 3} & -q_x \mathbf{Q}_i^T \\ \mathbf{0}_{1\times 3} & \mathbf{Q}_i^T & -q_y \mathbf{Q}_i^T \end{bmatrix}_{2\times 12} \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}_{12\times 1} = 0$$

Camera calibration

- Each pair **q** <--> **Q** provides 2 equations
- We have 12 unknowns but it's an homogeneous system...
- So we have 11 unknowns and we can set the last element to 1
- So... 2xN = 11
- N=5.5 points in general position are enough to estimate P

$$\begin{bmatrix} \mathbf{Q}_1^T & \mathbf{0}_{1\times 3} & -q_{x1}\mathbf{Q}_1^T \\ \mathbf{0}_{1\times 3} & \mathbf{Q}_1^T & -q_{y1}\mathbf{Q}_1^T \\ \vdots & \vdots & \vdots \\ \mathbf{Q}_N^T & \mathbf{0}_{1\times 3} & -q_{xN}\mathbf{Q}_N^T \\ \mathbf{0}_{1\times 3} & \mathbf{Q}_N^T & -q_{yN}\mathbf{Q}_N^T \end{bmatrix}_{2N\times 12} \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}_{12\times 1} = 0$$

Camera calibration

$$\begin{bmatrix} \mathbf{Q}_1^T & \mathbf{0}_{1\times3} & -q_{x1}\mathbf{Q}_1^T \\ \mathbf{0}_{1\times3} & \mathbf{Q}_1^T & -q_{y1}\mathbf{Q}_1^T \\ \vdots & \vdots & \vdots \\ \mathbf{Q}_N^T & \mathbf{0}_{1\times3} & -q_{xN}\mathbf{Q}_N^T \\ \mathbf{0}_{1\times3} & \mathbf{Q}_N^T & -q_{yN}\mathbf{Q}_N^T \end{bmatrix}_{2N\times12} \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}_{12\times1} = 0 \quad \rightarrow \quad \mathbf{A}_{2N\times12}\mathbf{x}_{12\times1} = \mathbf{0}_{12\times1}$$

- We need to solve an homogeneous linear system
- $\mathbf{0}$ is the trivial solution
- Minimize $|\mathbf{Ax}|^2=\mathbf{0}$ subject to $|\mathbf{x}|=1$
- How? SVD!

Camera calibration

$$\mathbf{A}_{2N\times 12}\mathbf{x}_{12\times 1} = \mathbf{0}_{12\times 1}$$

$$\mathbf{A}_{2N\times 12} \overset{\text{SVD}}{=} \mathbf{U}_{2N\times 12}\mathbf{D}_{12\times 12}\mathbf{V}^{T}_{12\times 12}$$

- The last column of $\mathbf{V}^T$ gives $\mathbf{x}$
- And $\mathbf{x}$ collects the columns of $\mathbf{P}$ stacked one on top of another

- This is called the **Direct Linear Transformation** (DLT)

Camera calibration

- Once we get $\mathbf{P}$ we remember that

$$\mathbf{P}_{3\times4} = \mathbf{K}_{3\times3}\left[\mathbf{R}_{3\times3} \mid \mathbf{t}_{3\times1}\right] = \left[\mathbf{KR} \mid \mathbf{Kt}\right]_{3\times4}$$

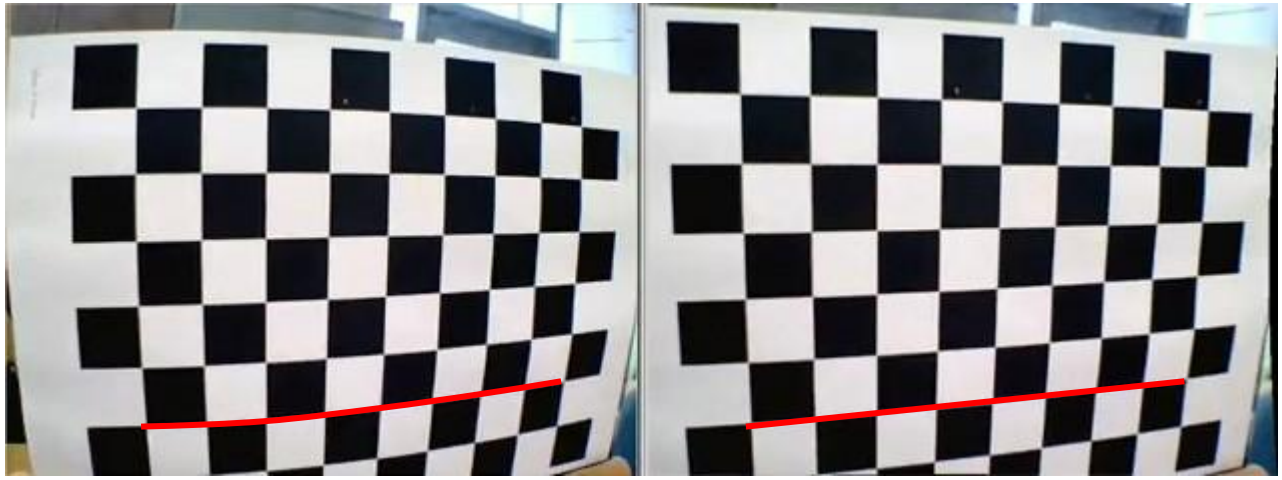
- Also **KR** is a 3x3 square matrix which is the product of a upper triangular matrix **K** and an orthogonal matrix **R**
- We can apply a RQ decomposition of this part of P to estimate **K** and **R**
- RQ decomposition:

Theorem 1. For any square real matrix $\mathbf{A}_{nxn}$ there exists a unique pair of a orthogonal matrix **Q** and an upper triangular matrix **R** with positive diagonal entries such that:

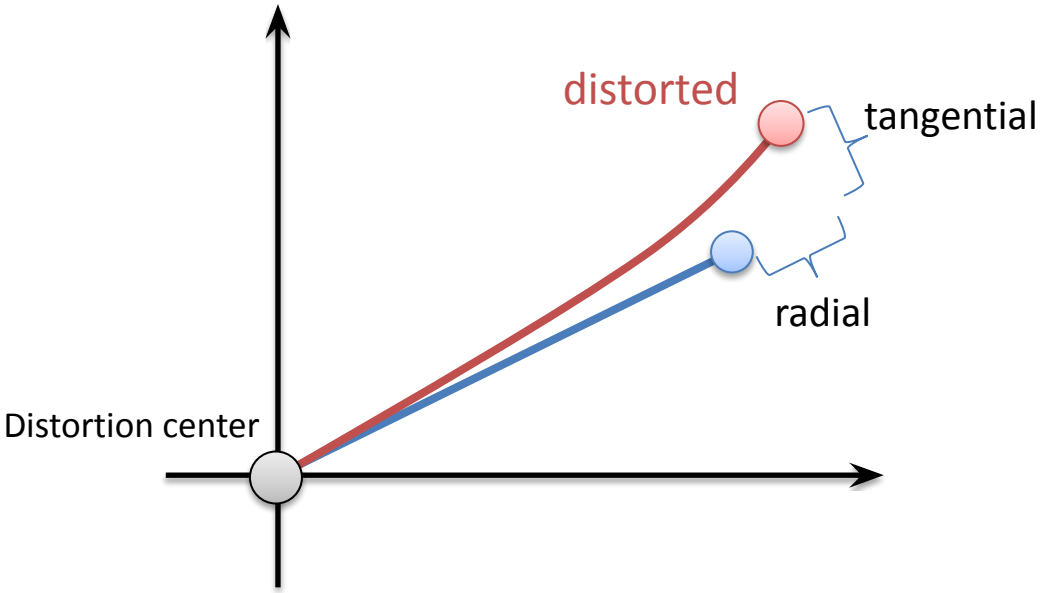
$$\mathbf{A} = \mathbf{Q}\,\mathbf{R}$$

Camera model – Optical distortion

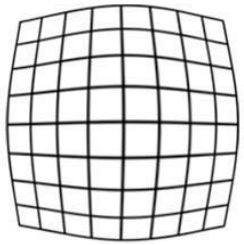
- Projection is a linear model
  - Accurate for pure pin-hole cameras
- Optical distortions due to the lens deviate from linearity
  - Bad quality lens
  - Fish-eyes or wide field of view lenses
- Distortion/Undistortion before applying calibration matrix
  - It applies to the projected points in camera coordinates



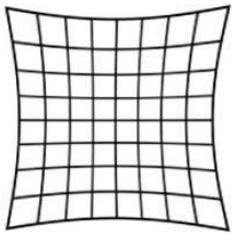
Camera model – Optical distortion



- Two main components
  - Radial, most important
  - Tangential, uncommon
- Assumption:
  - distortion center = principal point



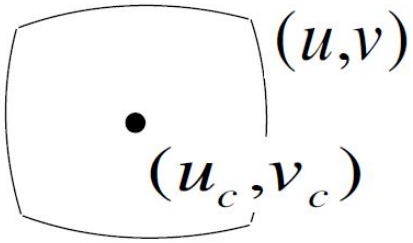
Barrel distortion: magnification decreases radially



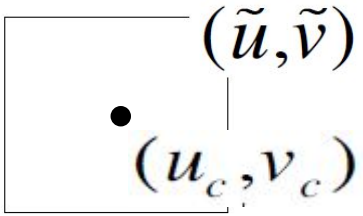
Pincushion distortion: magnification increases radially

Camera model – Optical distortion

Distorted coordinates



Undistorted coordinates



- The usual (radial) correction

$$\begin{cases} \tilde{u} = u_c + L(r)(u - u_c) \\ \tilde{v} = v_c + L(r)(v - v_c) \end{cases} \quad with \quad \begin{aligned} r^2 &= (u - u_c)^2 + (v - v_c)^2 \\ L(r) &= 1 + k_1 r + k_2 r^2 + k_3 r^3 \end{aligned}$$

- The parameters k_i estimated during calibration
- Correction
  - Undistort points (opencv `undistortPoints()`)
  - Distort points when projecting 3D points (opencv `projectPoints()`)

Camera calibration in practice

- The full projection model (with distortion) is not linear

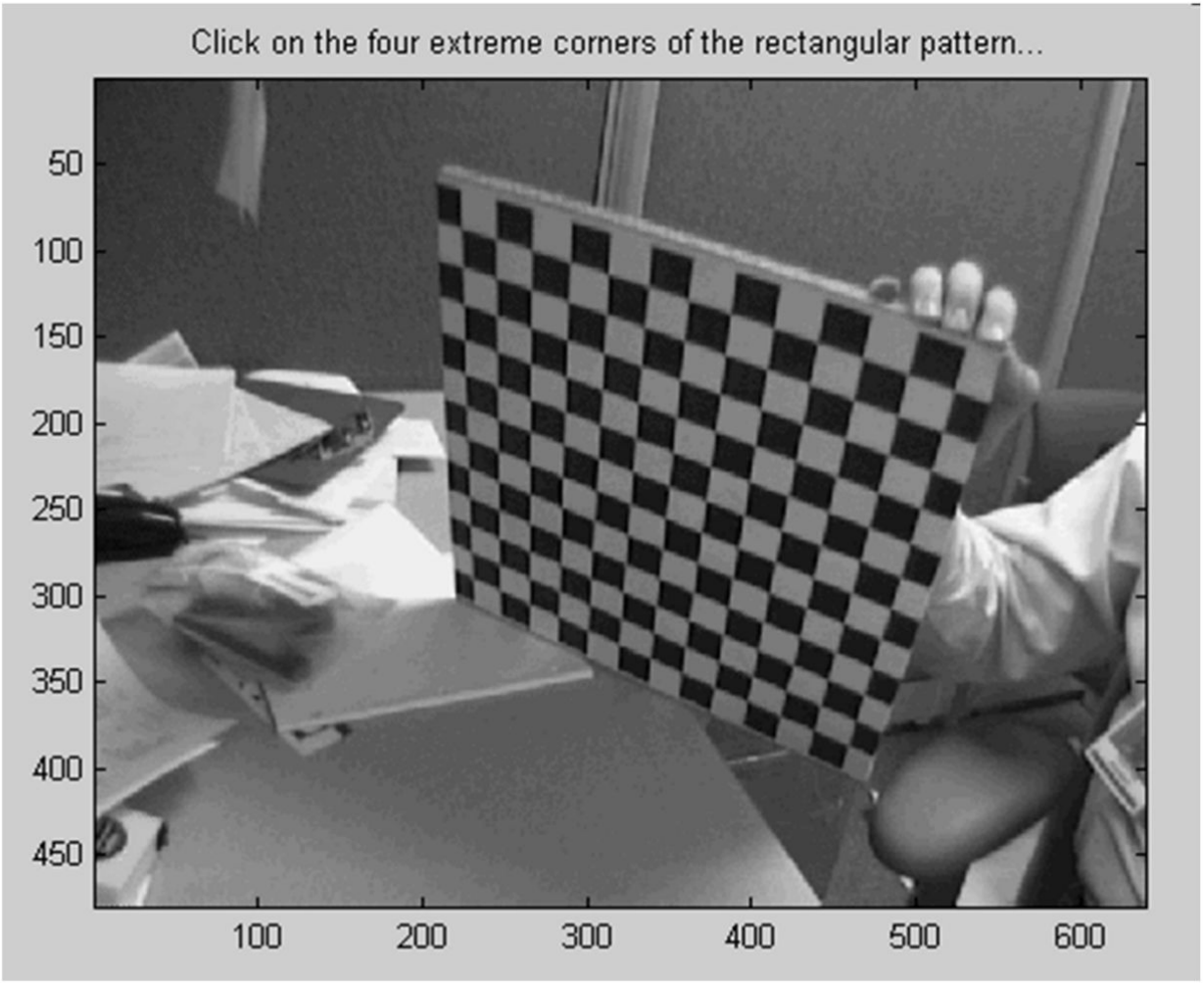
$$\mathbf{q} \sim \mathbf{K}\, f_{\text{dist}}\left(\left[\mathbf{R}\,|\,\mathbf{t}\right]\mathbf{Q}\right)$$

- Where f_{dist} is the distortion model to apply
- Camera calibration then should estimate
  - Intrinsics parameters (K)
  - Distortion parameters (radial, tangential etc)
  - External parameters (R and t)
- Normally tackled in two steps:
  - 1. first discard distortion and use DLT method to estimate K, R, and t
  - 2. re-estimate the full model with distortion through non-linear optimization.

$$\min \sum_i d\left(\mathbf{q}_i, \mathbf{K} f_{\text{dist}}\left(\left[\mathbf{R}\,|\,\mathbf{t}\right]\mathbf{Q}_i\right)\right)$$

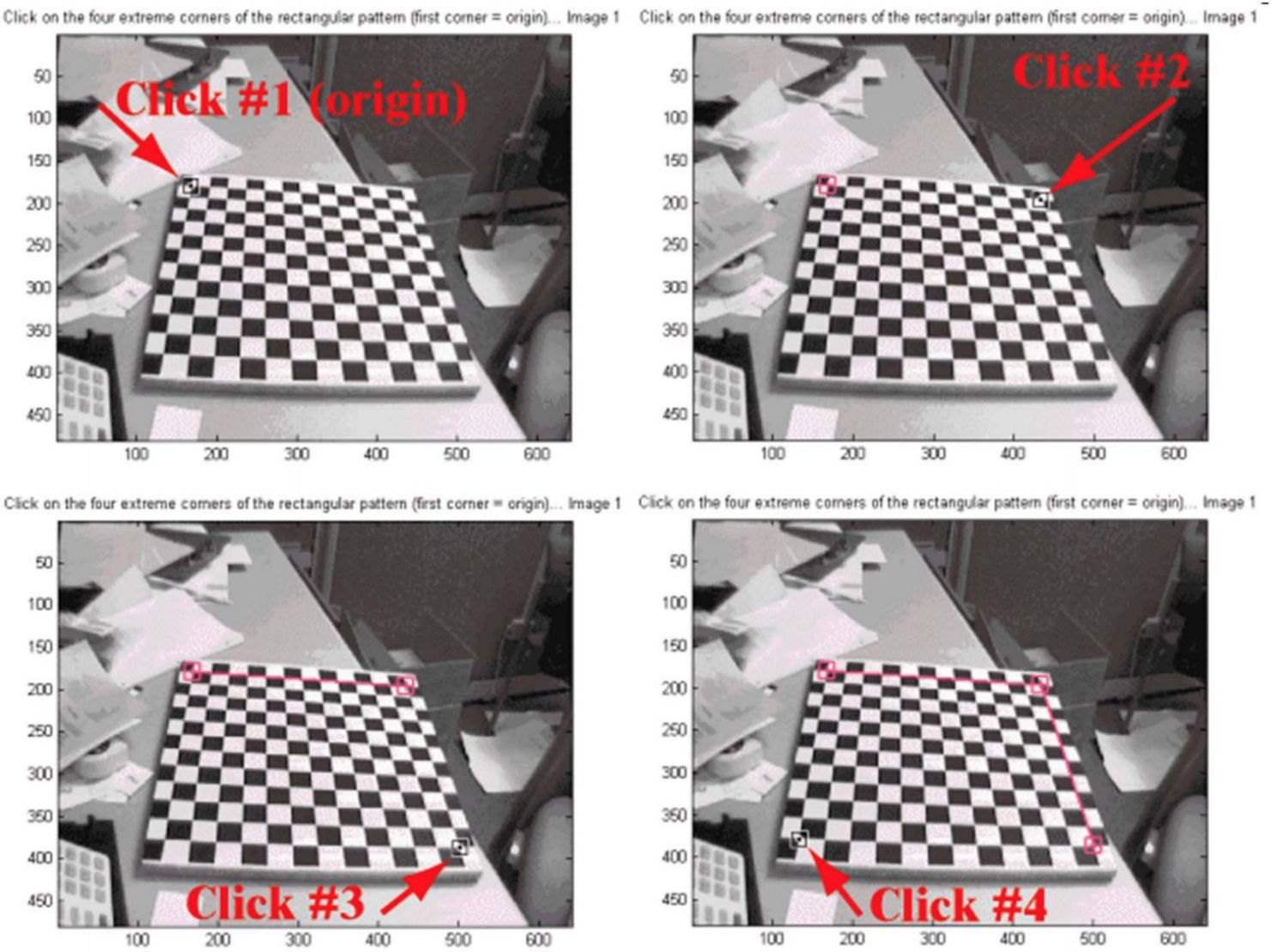
Camera calibration in practice

Camera Calibration Toolbox for Matlab J. Bouguet – [1998-2000]

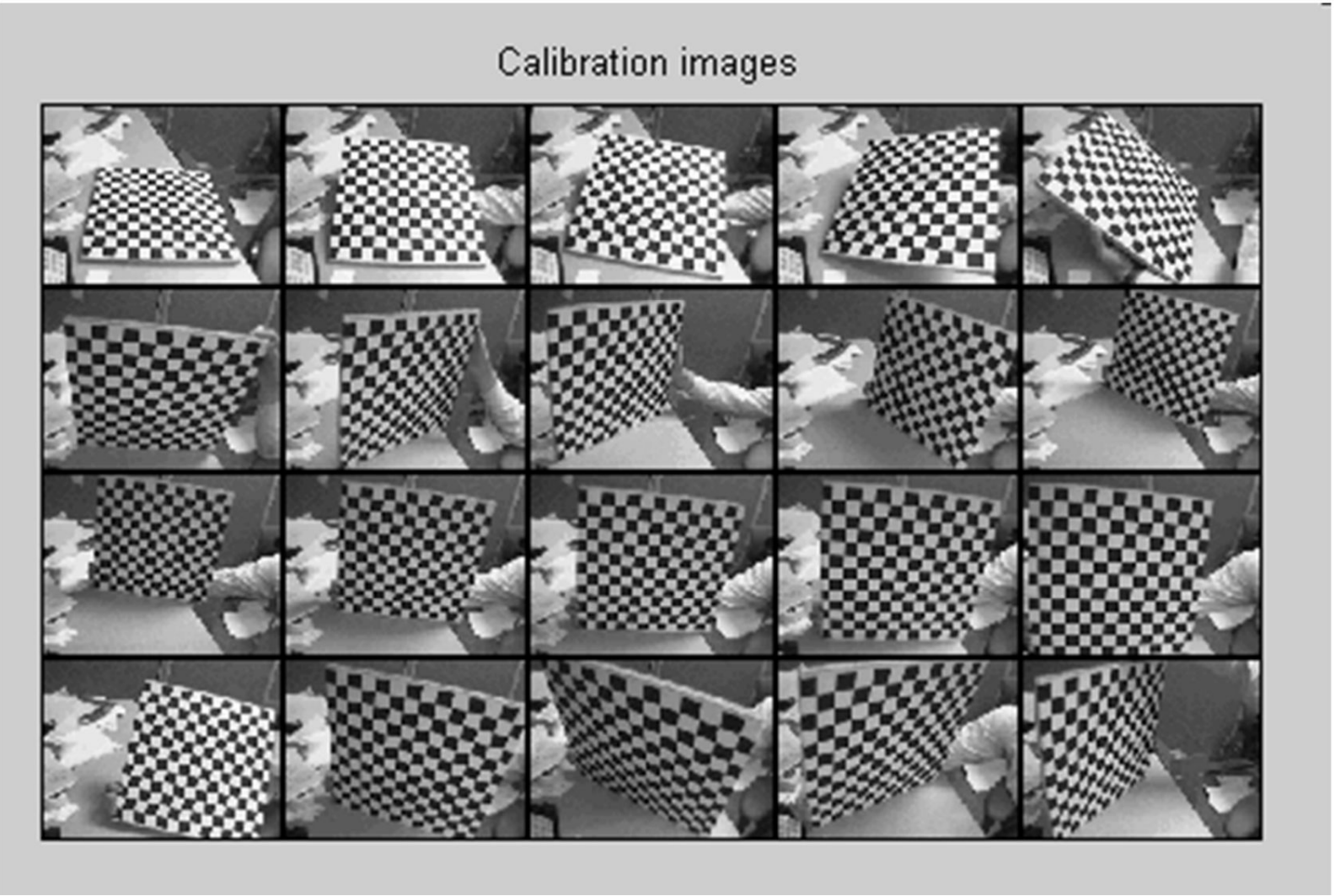


http://www.vision.caltech.edu/bouguetj/calib_doc/index.html#examples

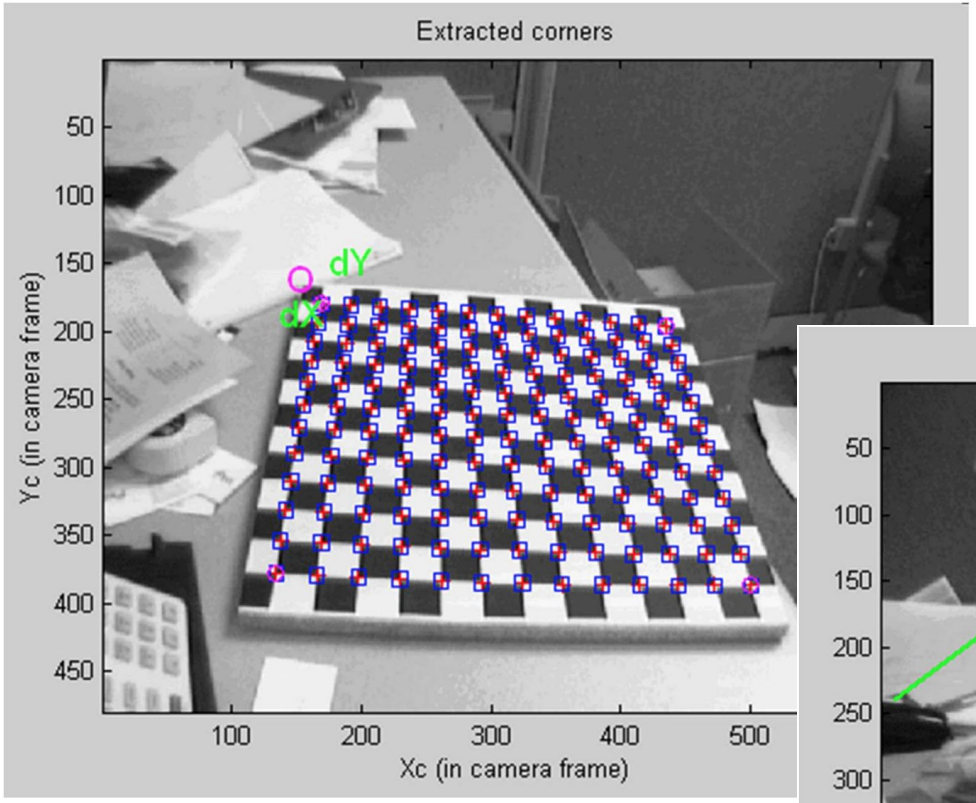
Camera calibration in practice

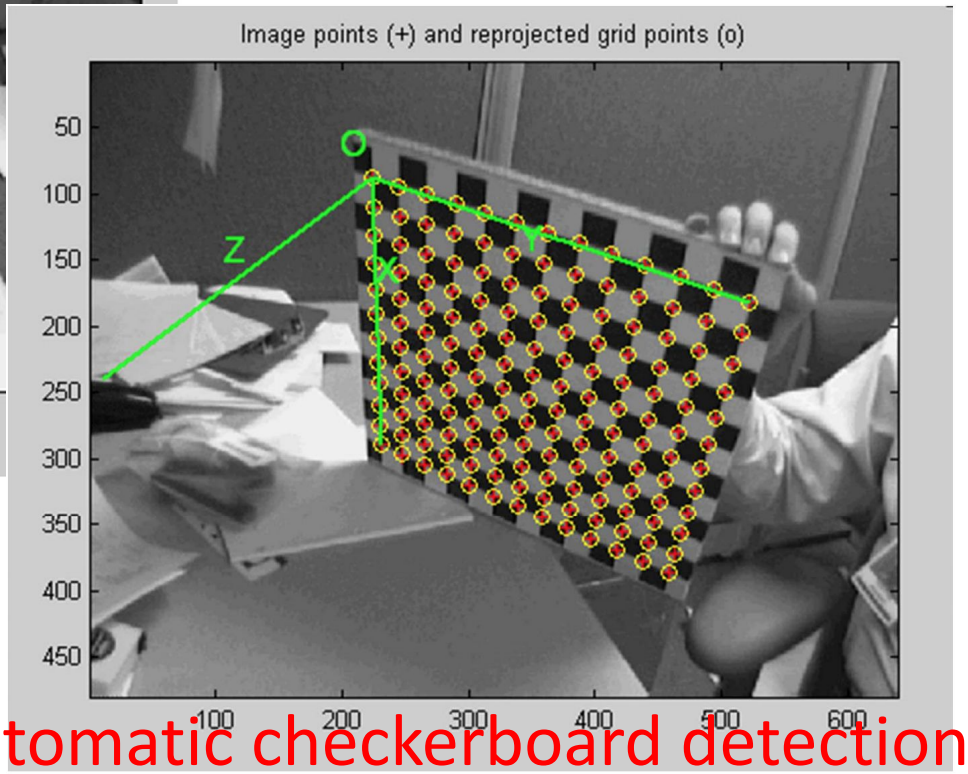


Camera calibration in practice



Camera calibration in practice





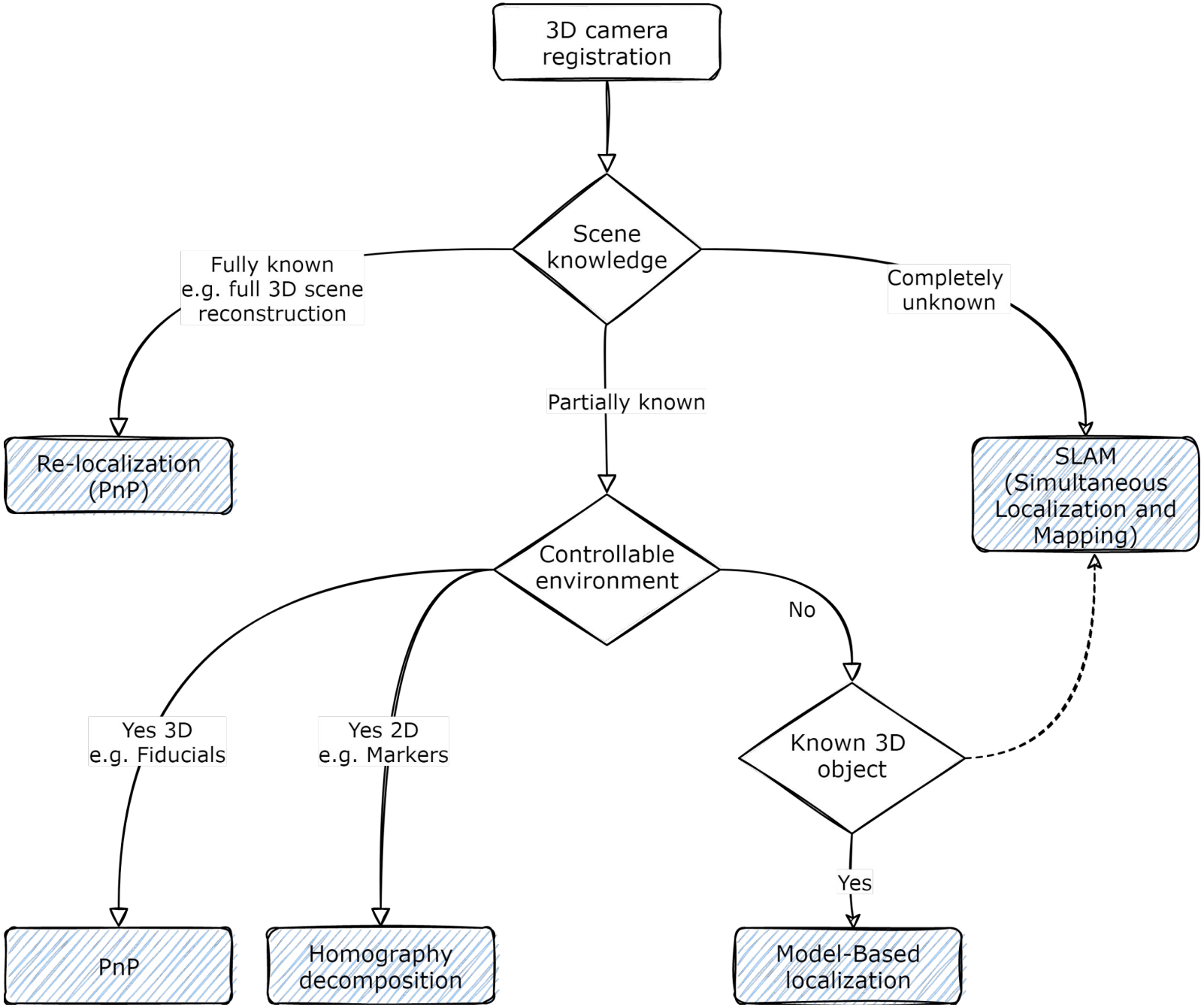
Ported to opencv with automatic checkerboard detection

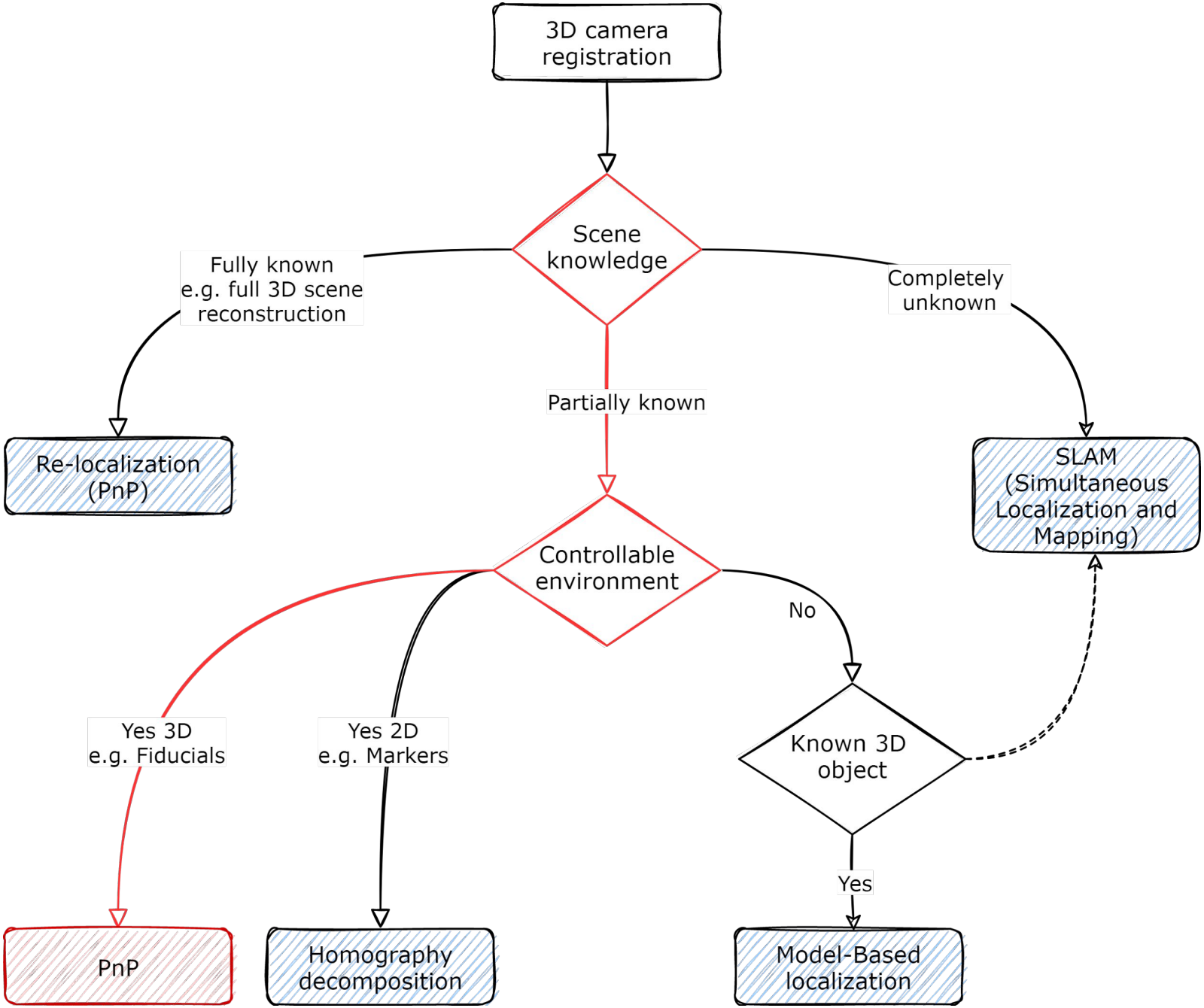
Today Topics

- The registration problem
- Optical visual tracking for AR
  - Background – camera model and calibration
  - Pose estimation using fiducials
    - Model Based tracking
  - Pose estimation using markers

The registration problem

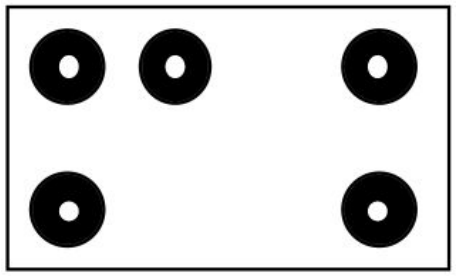
- Assumptions about the scene (What do we know?)
  - Some 3D references in the scene (fiducials)
  - The 3D model of an object to track (model-based tracking)
  - Some 2D reference in the scene (markers)
  - Nothing? Use the natural features (SLAM)


3D camera registration
Scene knowledge
Fully known e.g. full 3D scene reconstruction
Completely unknown
Partially known
Re-localization (PnP)
SLAM (Simultaneous Localization and Mapping)
Controllable environment
No
Yes 3D e.g. Fiducials
Yes 2D e.g. Markers
Known 3D object
Yes
PnP
Homography decomposition
Model-Based localization


3D camera registration
Scene knowledge
Fully known e.g. full 3D scene reconstruction
Completely unknown
Partially known
Re-localization (PnP)
SLAM (Simultaneous Localization and Mapping)
Controllable environment
No
Yes 3D e.g. Fiducials
Yes 2D e.g. Markers
Known 3D object
Yes
PnP
Homography decomposition
Model-Based localization

Fiducial points in the scene

- Insert “artificial” points in the scene
  - easily detectable
  - with known, measured (absolute) position
- Laser used to measure exact location of each marker wrt a world reference system
- The scene needs to be set up
- Circular marker are preferred as it’s easy to detect their centre



Spatial combination allows identification

Different patterns allow identification

Circular Markers CCTag

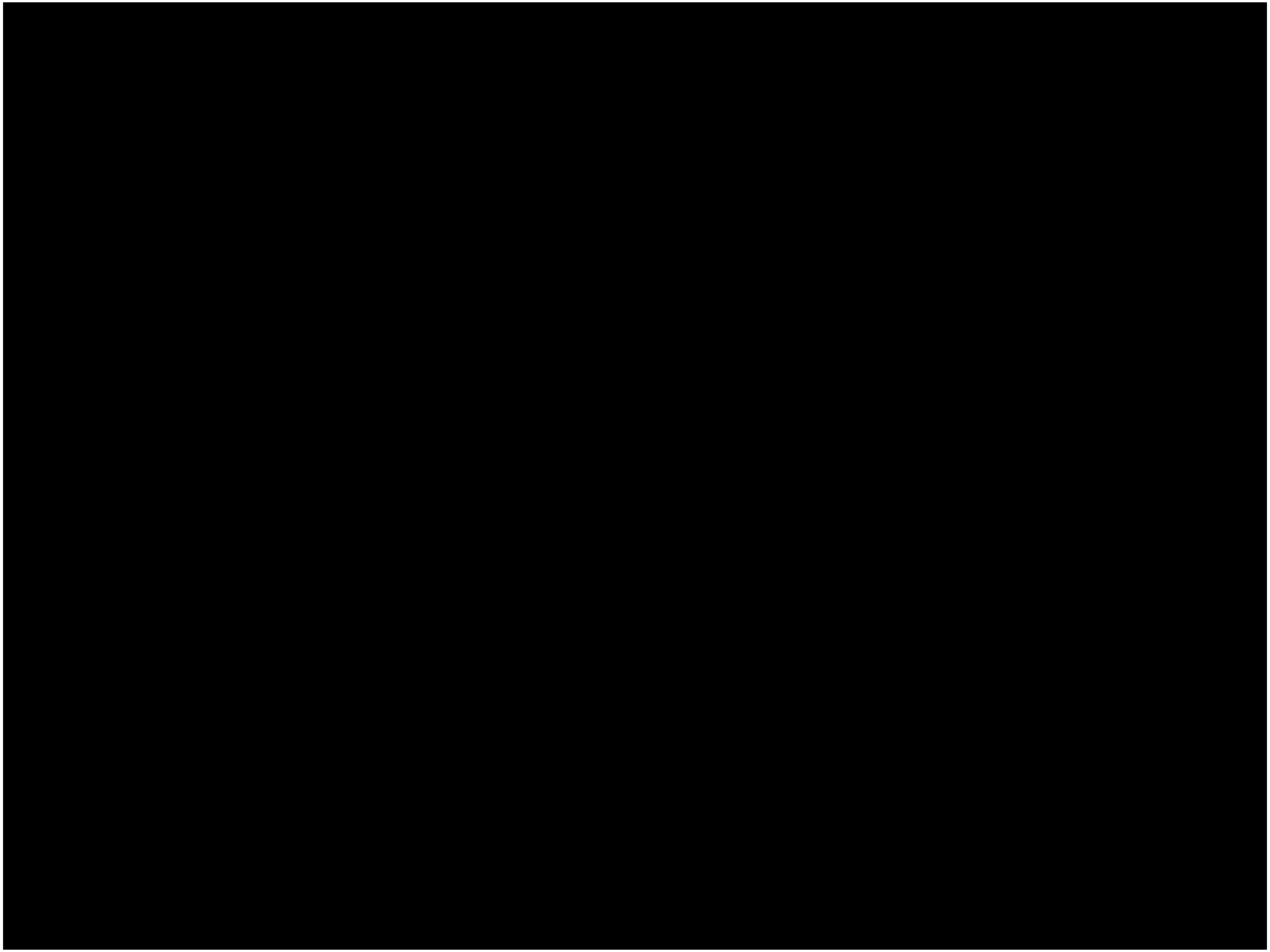
- Normally used in photogrammetry to detect a single point with high accuracy (fiducials)
  - More pixels are used to fit the ellipse on the image
  - The image of the center is estimated in a more robust way
- Work by Lilian Calvet @IRIT, eq. REVA N7
  - CCTag, concentric circles
  - Allows to estimate the pose with at least 4 markers
  - Real time on GPU

Fork me on GitHub

CCTAG

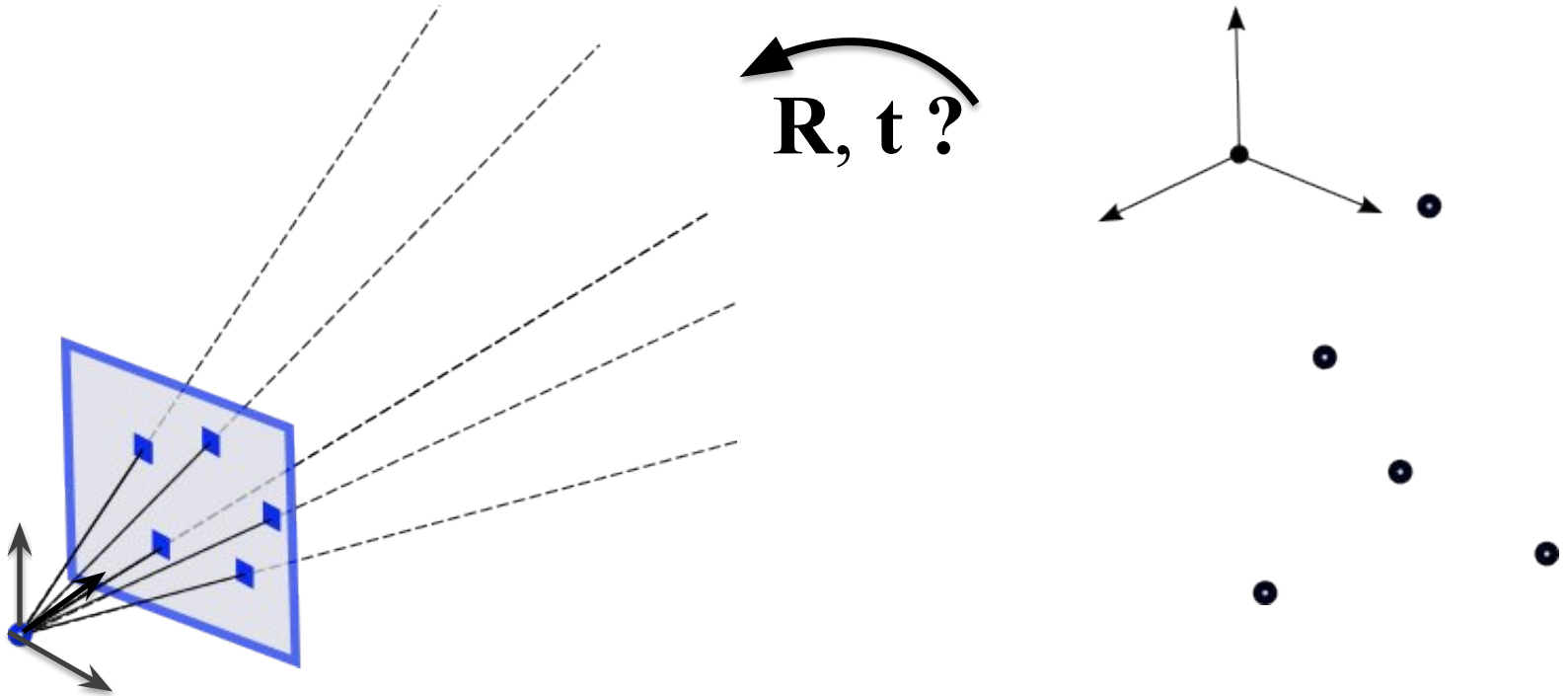
https://github.com/alicevision/CCTag

Fast and accurate CPU-GPU marker detection



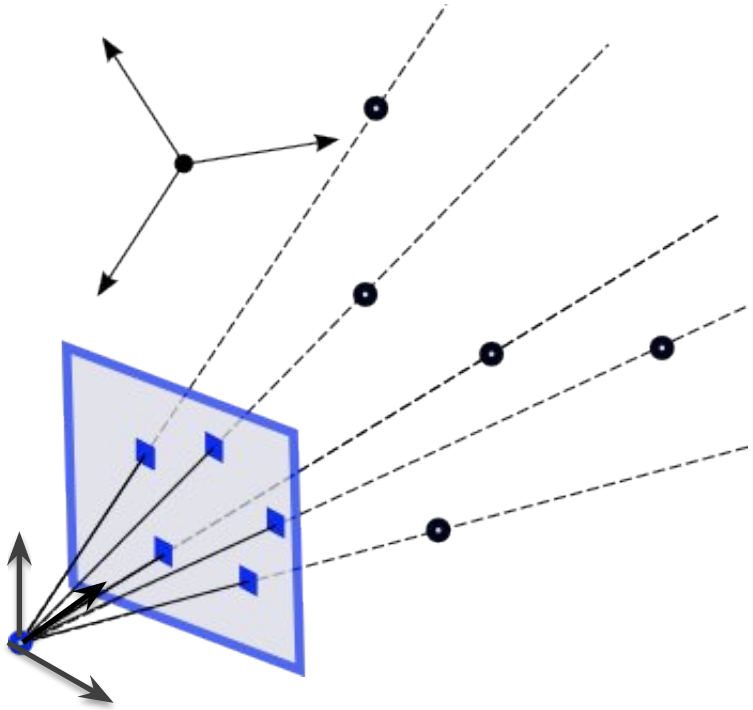
Pose estimation

- Given a set of 3D points expressed in some world reference system
- Their corresponding image points
- Estimate the rototranslation $[\mathbf{R},\mathbf{t}]$ that "aligns" the 3D points



Pose estimation

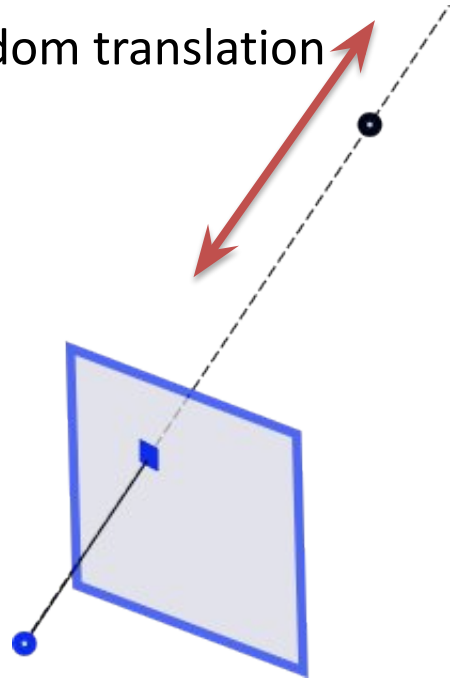
- Given a set of 3D points expressed in some world reference system
- Their corresponding image points
- Estimate the rototranslation [R,t] that “aligns” the 3D points



Perspective-n-Points problem (PnP)

- How many correspondences are needed?

1 degree of freedom translation
3 d.o.f rotations

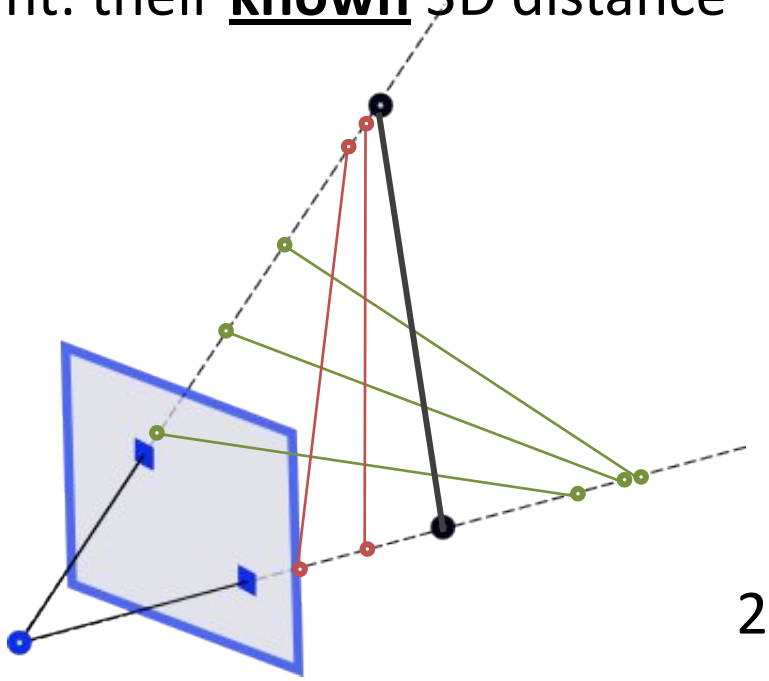


1 is not enough

PnP problem

- How many correspondences are needed?

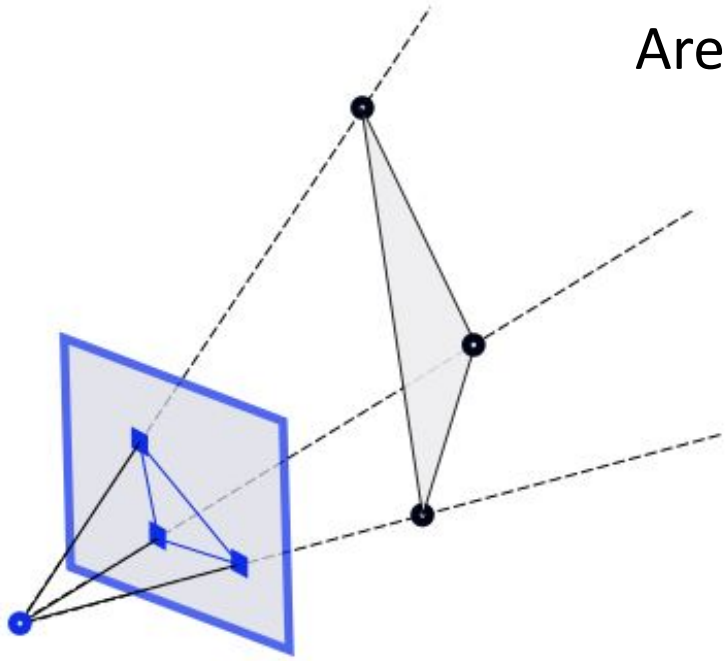
With 2 points we introduce a new constraint: their **known** 3D distance



2 are still not enough

PnP problem

- How many correspondences are needed?

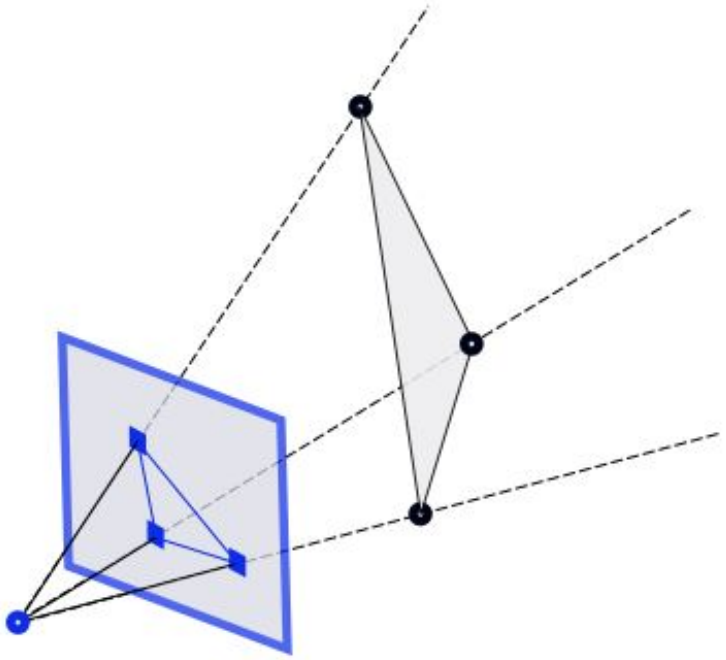


Are 3 enough??

?

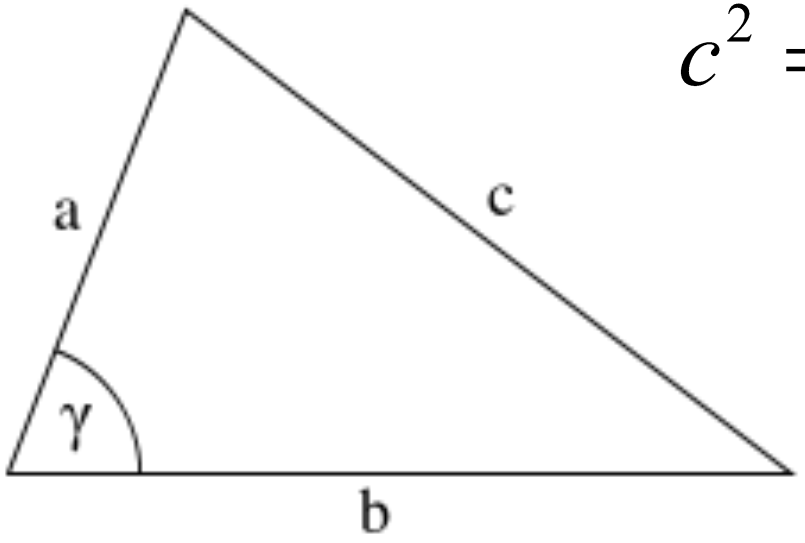
PnP problem

- How many correspondences are needed?
- In general it can be proved that 3 points can give **at most 4 possible solutions**
  - 1 or more points can be used to disambiguate



P3P problem

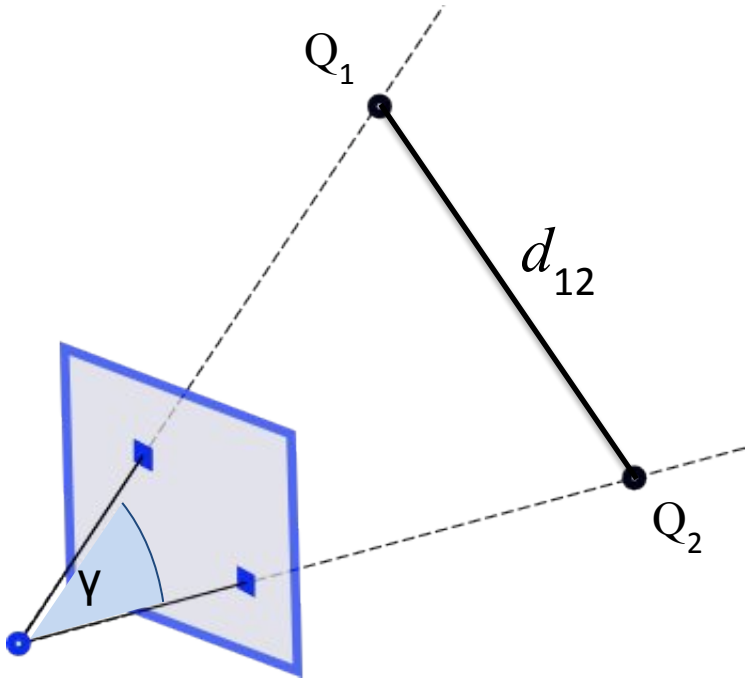
- Remember the law of cosines? (“théorème d'Al-Kashi”)



$$c^2 = a^2 + b^2 - ab\cos\gamma$$

P3P problem

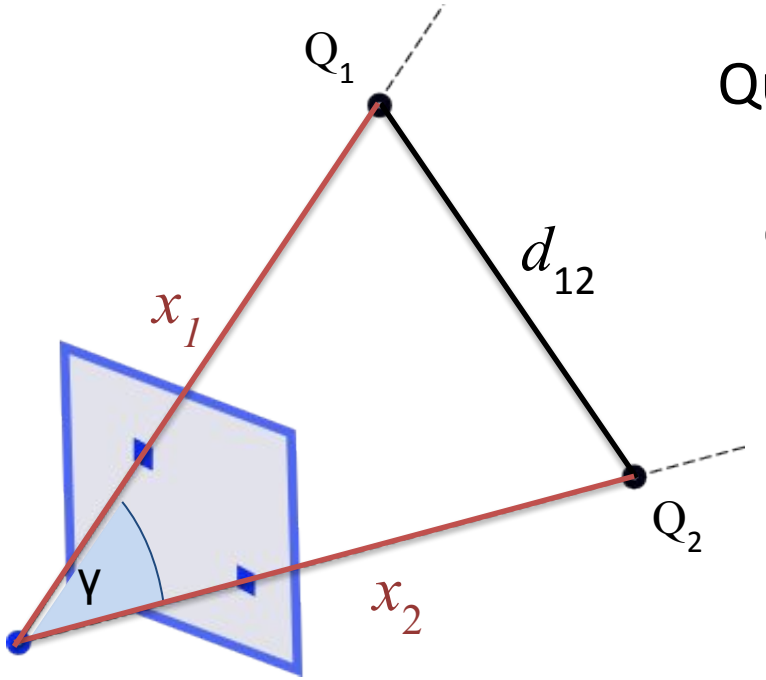
- Main idea:



P3P problem

- Main idea:

$$d_{12}^2 = x_1^2 + x_2^2 - x_1 x_2 \cos\gamma$$



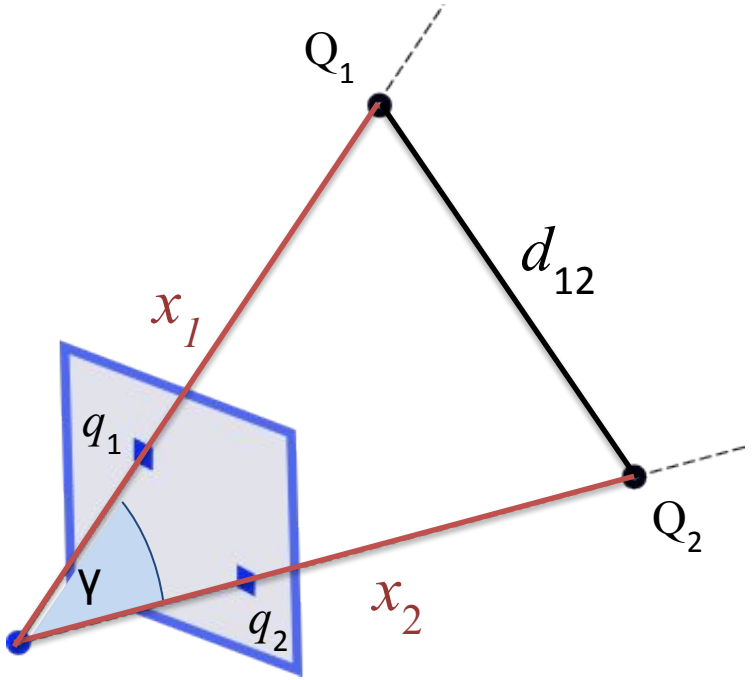
Quadratic equation in 2 **unknowns** x_1 and x_2

Given: γ and d_{12}

P3P problem

- Main idea:

$$d_{12}^2 = x_1^2 + x_2^2 - x_1 x_2 \cos\gamma$$



Quadratic equation in x_1 and x_2

Given: γ and d_{12}

The angle between 2 rays:

$$\cos\gamma = \frac{q_1^T \omega q_2}{\sqrt{q_1^T \omega q_1}\sqrt{q_2^T \omega q_2}}$$

$$\omega = \left(\mathbf{K}\mathbf{K}^T\right)^{-1} = \mathbf{K}^{-T}\mathbf{K}^{-1}$$

The image of the absolute conic

P3P problem

Solve the quadratic system in x1, x2, x3

$$\begin{cases} d_{12}^2 = x_1^2 + x_2^2 - x_1 x_2 \cos\gamma \\ d_{23}^2 = x_2^2 + x_3^2 - x_2 x_3 \cos\beta \\ d_{13}^2 = x_1^2 + x_3^2 - x_1 x_3 \cos\alpha \end{cases}$$

Up to 4 solutions [Grunert 1841]

P3P problem

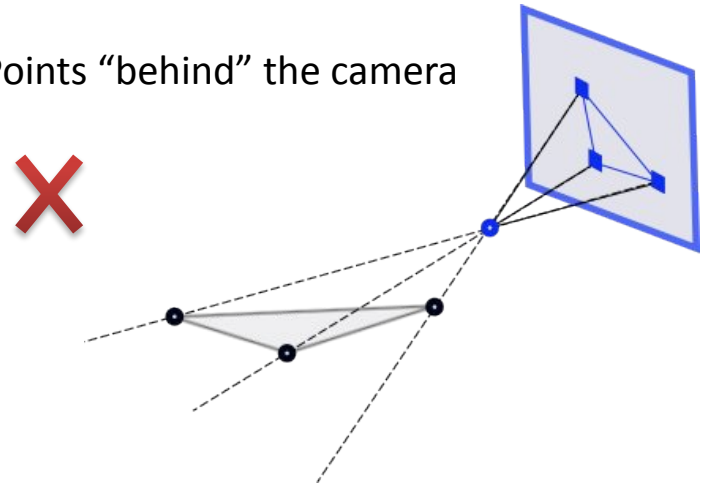
Solve the quadratic system in x1, x2, x3

$$\begin{cases} d_{12}^2 = x_1^2 + x_2^2 - x_1 x_2 \cos\gamma \\ d_{23}^2 = x_2^2 + x_3^2 - x_2 x_3 \cos\beta \\ d_{13}^2 = x_1^2 + x_3^2 - x_1 x_3 \cos\alpha \end{cases}$$

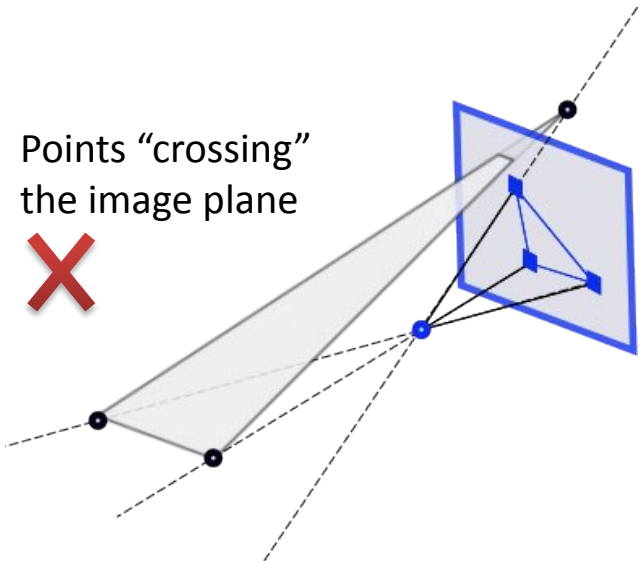
Up to 4 solutions [Grunert 1841]

- Some of them are not physically feasible…

Points “behind” the camera

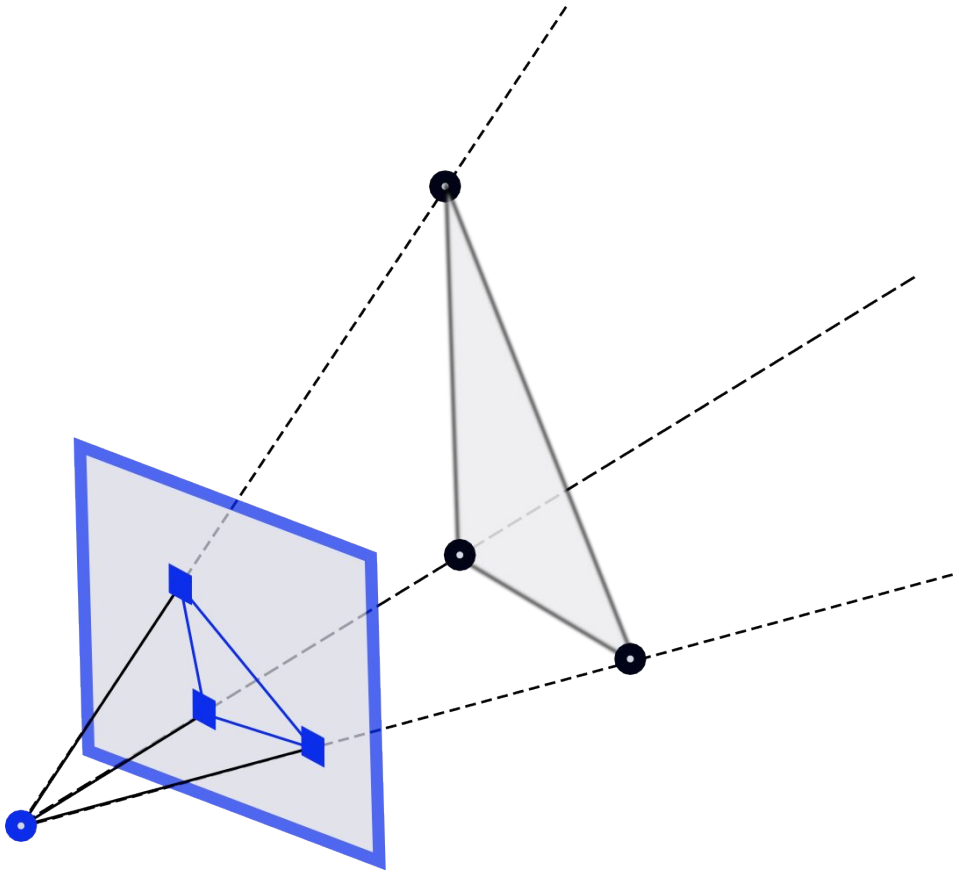


Points “crossing” the image plane



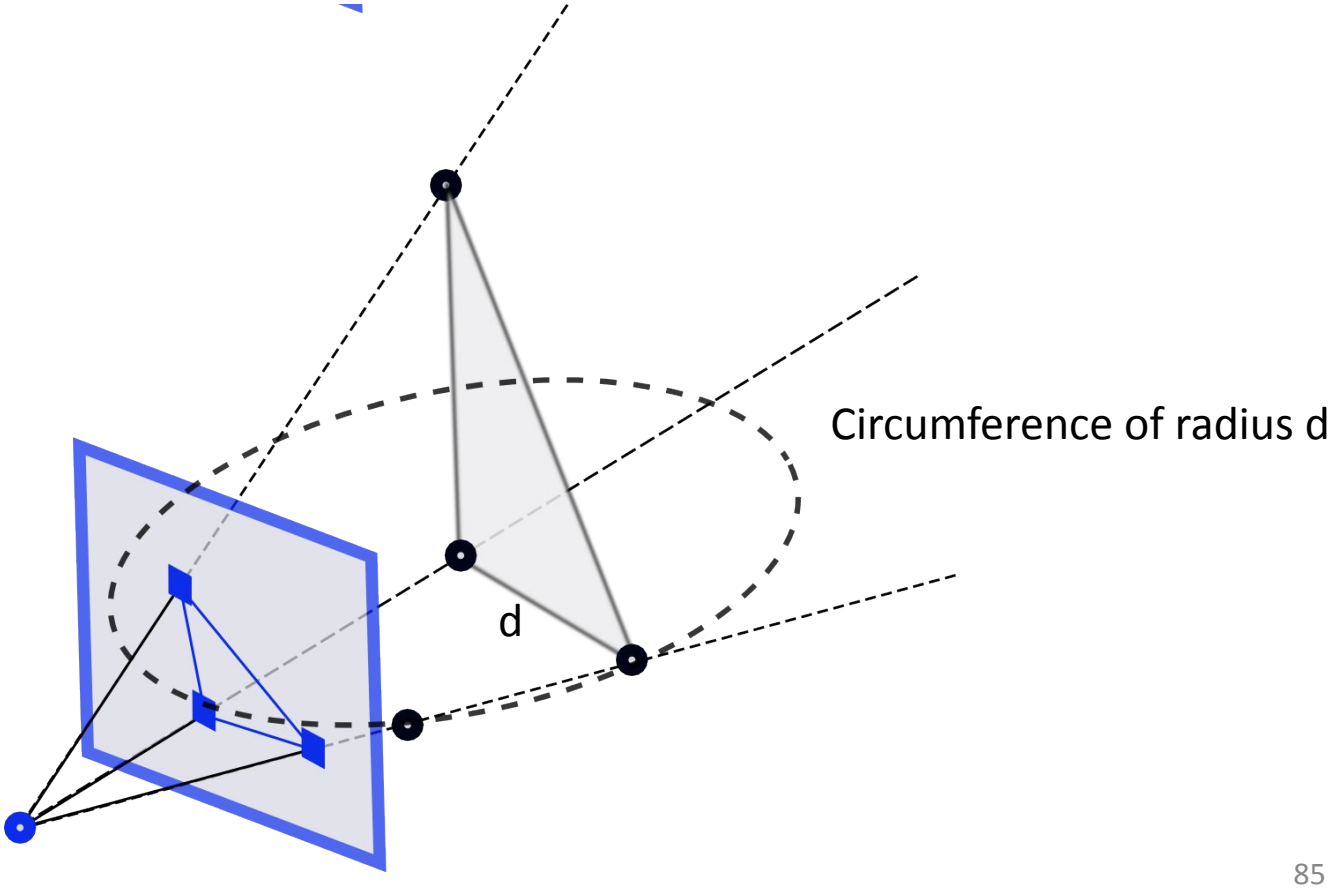
P3P

- There may be ambiguities
  - 2 or more possible (physically feasible) solutions



P3P

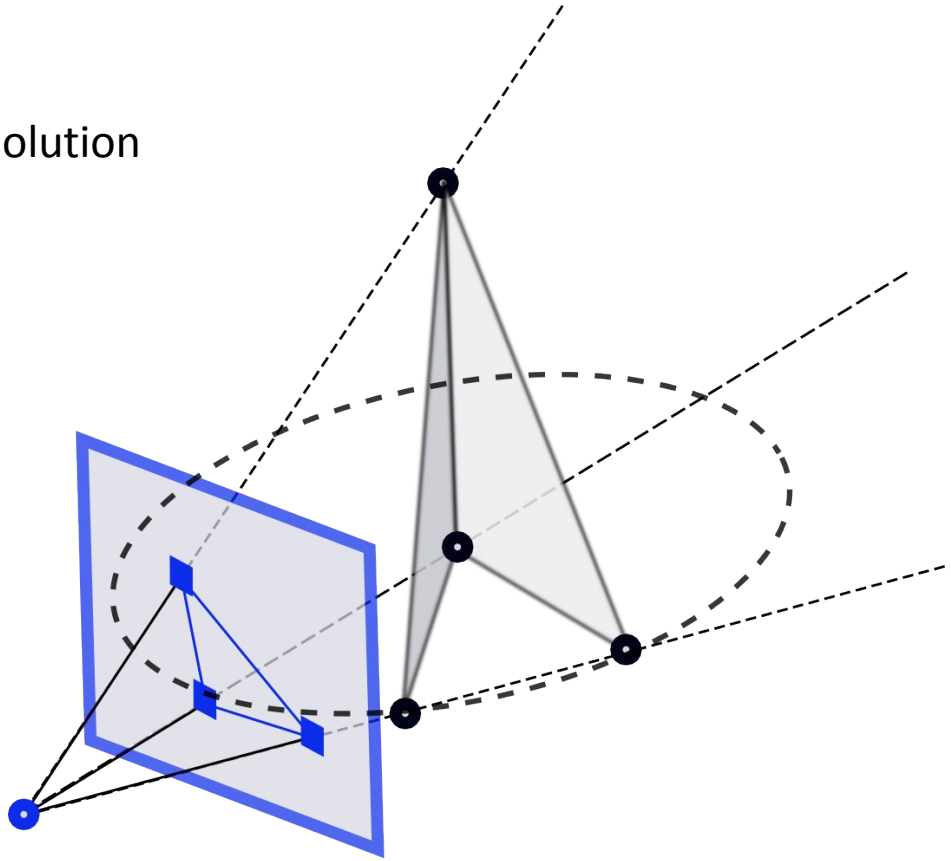
- There may be ambiguities
  - 2 or more possible (physically feasible) solutions



P3P

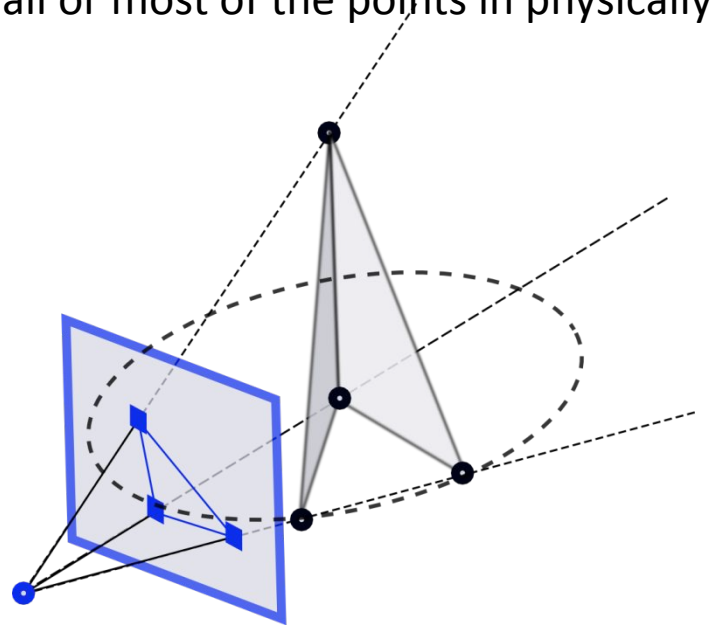
- There may be ambiguities
  - 2 or more possible (physically feasible) solutions

A second valid solution



P3P

- Minimal solver
- There may be ambiguities
  - 2 or more possible (physically feasible) solutions
- Use a 4^{th} or eventually more points to disambiguate
  - Use RANSAC to generate a solution and then evaluate:
    - The reprojection error
    - Solution with all or most of the points in physically feasible solution



Robust estimation with Ransac

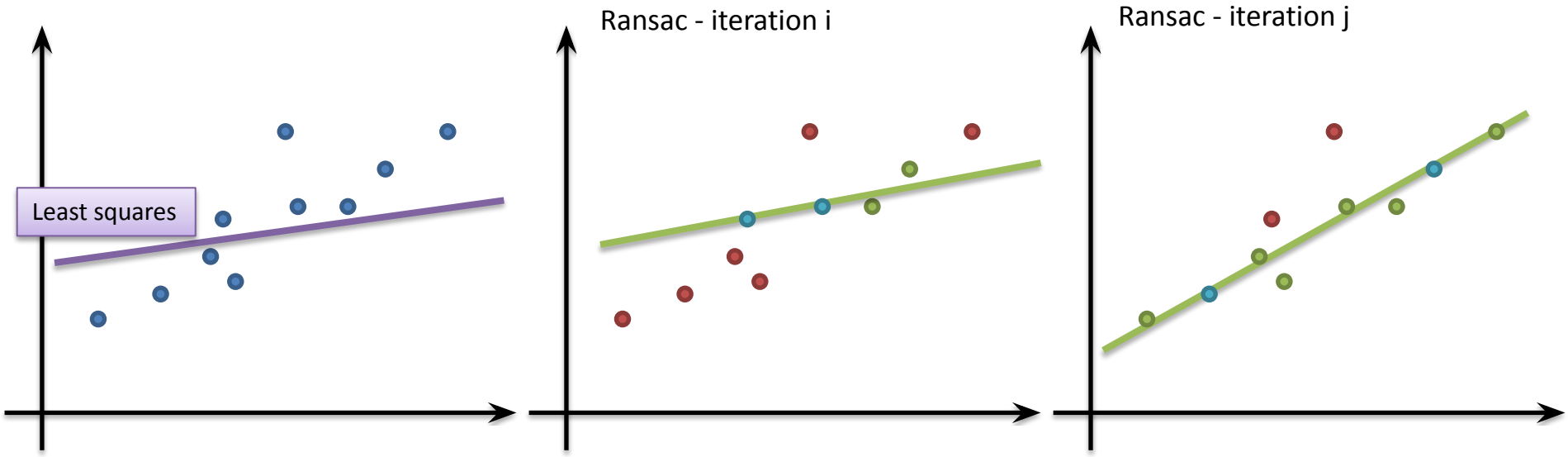
RANSAC (RANdom SAmpling Consensus)

- General framework to model estimation/fitting in presence of noise, ie outliers
- Main idea:
  - Select a sample of data enough to generate a model
    - i.e. use a minimal solver
  - Compare all the data against the model
    - Compute the distance of the data from the model
  - Get two sub-sets, inliers and outliers
  - Iterate N times and keep the best model and its inliers
  - Refine model using only the inliers

M. A. Fischler, R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. of the ACM, Vol 24, pp 381-395, 1981.

Robust estimation with Ransac

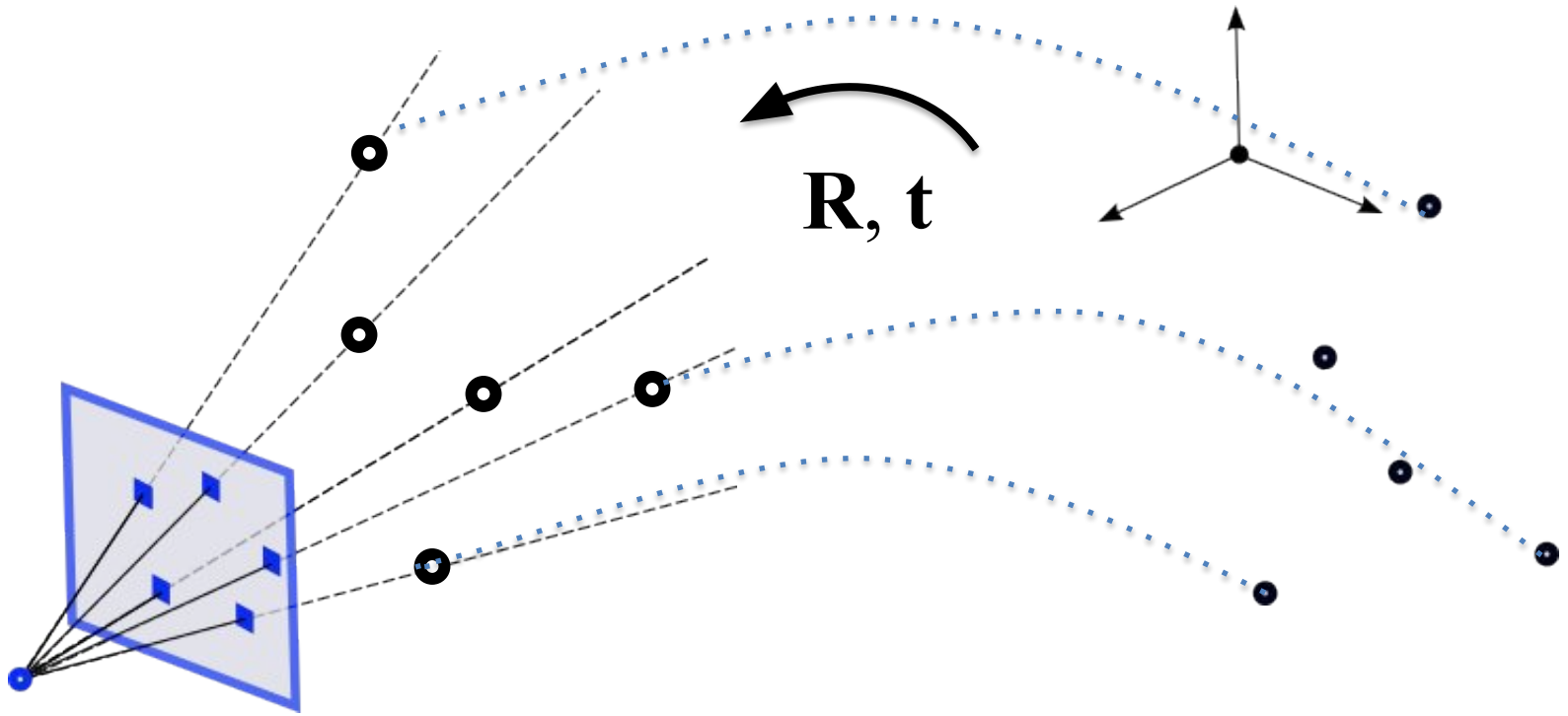
- RANSAC and line fitting:
- Repeat **N** times:
  - Draw **s** points uniformly at random
  - Fit line to these s points
  - Find inliers to this line among the remaining points (i.e., points whose distance from the line is less than **t**)



PnP

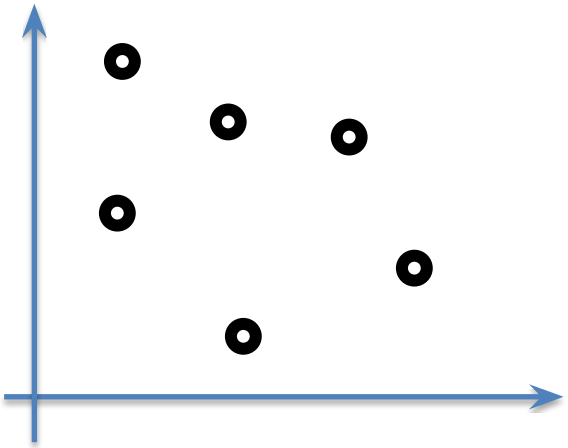
Estimation of R and t

- Once we have computed the d_{ij}
- Estimate the rototranslation [R t] from 3D-3D correspondences

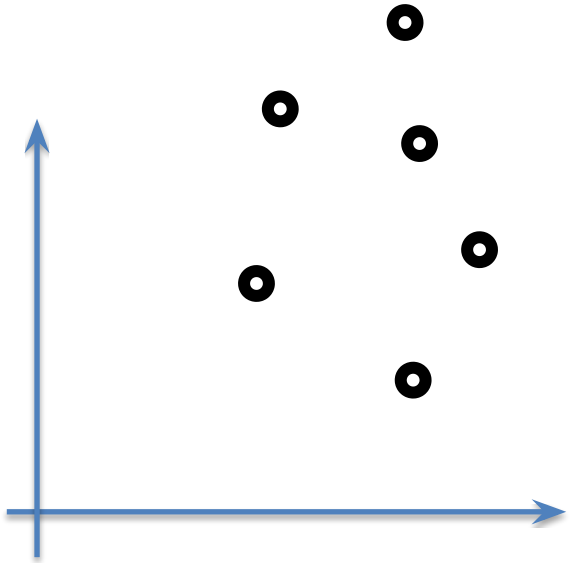


PnP

Estimation of R and t



3D points in the camera reference frame

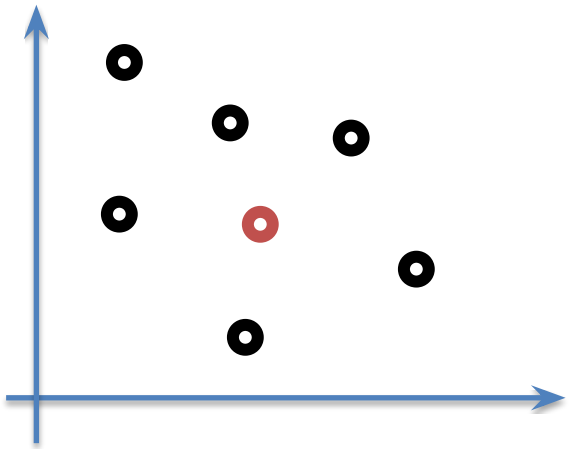


3D points in the world reference frame

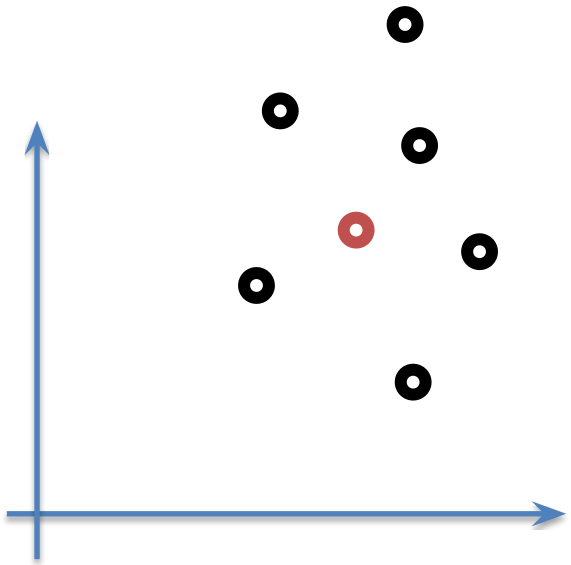
PnP

Estimation of R and t

Estimate the **mean point** (center of mass) of each set



3D points in the camera reference frame

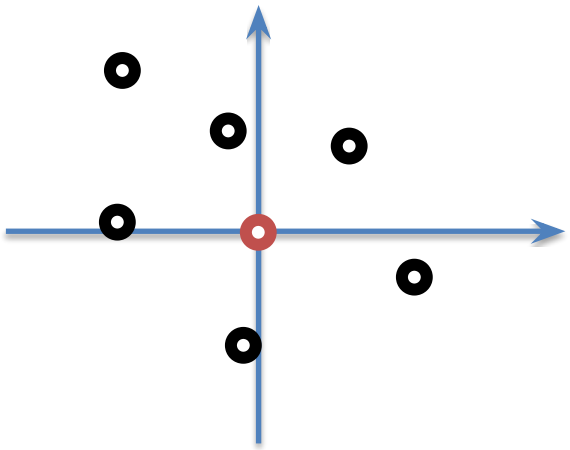


3D points in the world reference frame

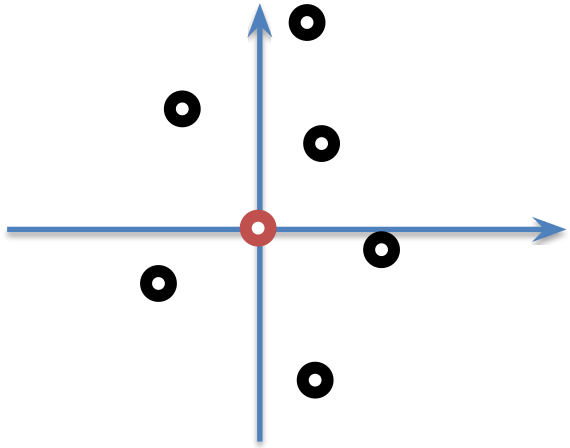
PnP

Estimation of R and t

Move the reference systems so that the **mean points** are the respective origins



3D points in the camera reference frame

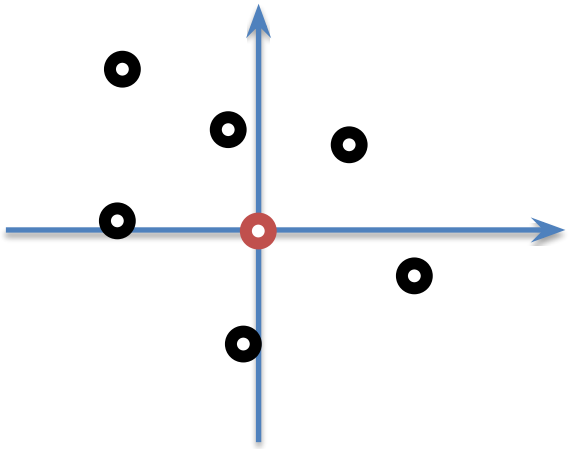


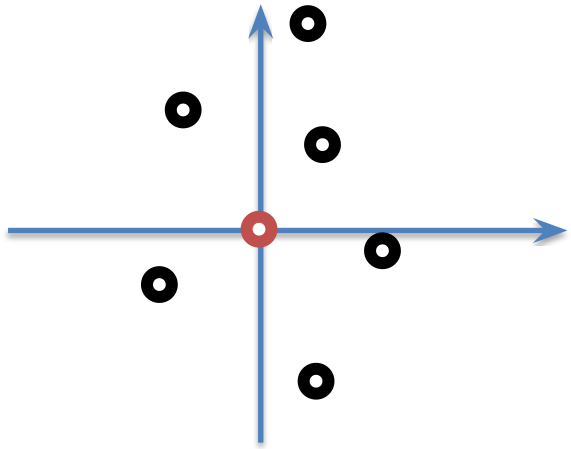
3D points in the world reference frame

PnP

Estimation of R and t

Now the two sets of points only differs by a 3D rotation around their center of mass





3D points in the camera reference frame

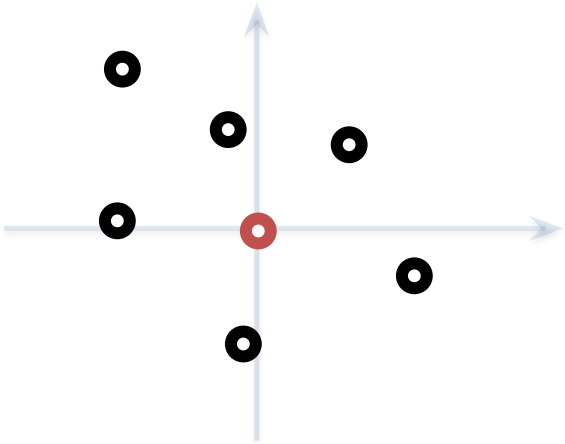
3D points in the world reference frame

Barycentric coordinates

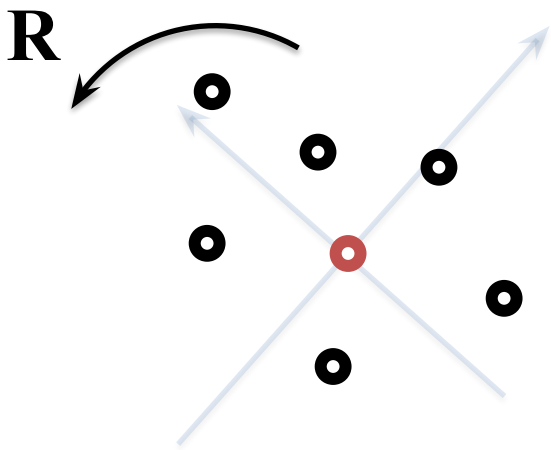
PnP

Estimation of R and t

Now the two sets of points only differs by a 3D rotation around their center of mass



3D points in the camera reference frame

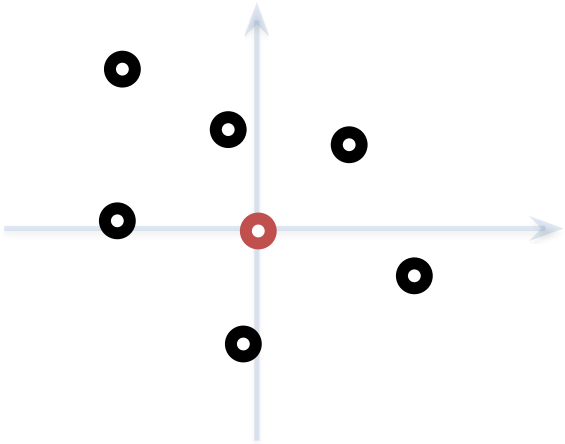


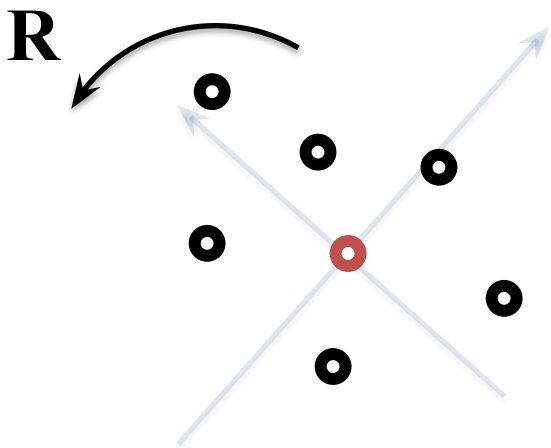
3D points in the world reference frame

PnP

Estimation of R and t

Now the two sets of points only differs by a 3D rotation around their center of mass





3D points in the camera reference frame

$\mathbf{Q}'_{c\,3\times N}$ 3D points in barycentric coordinates

3D points in the world reference frame

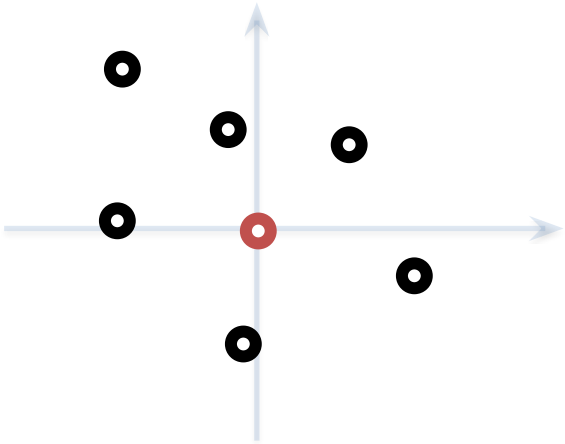
$\mathbf{Q}'_{w3\times N}$ 3D points in barycentric coordinates

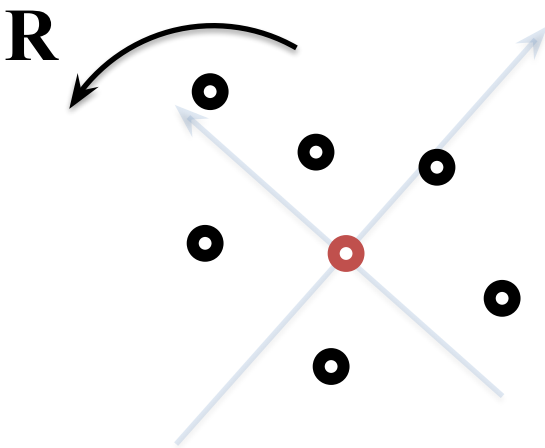
$$\mathbf{Q}'_{c\,3\times N} = \mathbf{R}\,\mathbf{Q}'_{w3\times N}$$

PnP

Estimation of R and t

Now the two sets of points only differs by a 3D rotation around their center of mass





3D points in the camera reference frame

$\mathbf{Q}'_{c\,3\times N}$ 3D points in barycentric coordinates

3D points in the world reference frame

$\mathbf{Q}'_{w3\times N}$ 3D points in barycentric coordinates

$$\mathbf{Q}'_{c\,3\times N} = \mathbf{R}\,\mathbf{Q}'_{w3\times N}$$

Solve as usual with pseudo-inverse + SVD for constraining R to be orthogonal

PnP

$$\mathbf{Q}'_{c\,3\times N} = \mathbf{R}\,\mathbf{Q}'_{w\,3\times N}$$

Multiply by $\mathbf{Q}'^{\,T}_{w\,N\times 3}$

$$\mathbf{Q}'_{c\,3\times N}\mathbf{Q}'^{\,T}_{w\,N\times 3} = \mathbf{R}\,\mathbf{Q}'_{w\,3\times N}\mathbf{Q}'^{\,T}_{w\,N\times 3}$$

Multiply by $\left(\mathbf{Q}'_{w\,3\times N}\mathbf{Q}'^{\,T}_{w\,N\times 3}\right)^{-1}$

$$\mathbf{Q}'_{c\,3\times N}\left[\mathbf{Q}'^{\,T}_{w\,N\times 3}\left(\mathbf{Q}'_{w\,3\times N}\mathbf{Q}'^{\,T}_{w\,N\times 3}\right)^{-1}\right] = \mathbf{R}$$

Pseudo-inverse of $\mathbf{Q}'_{w\,3\times N}$

$$\mathbf{A} = \mathbf{X}\mathbf{B} \rightarrow \mathbf{X} = \mathbf{A}\mathbf{B}^T\left(\mathbf{B}\mathbf{B}^T\right)^{-1}$$

Due to noise, in general, **R** won't be orthogonal. Take the "closest" orthogonal (in the sense of Frobenius norm) matrix using the SVD decomposition

$$\mathbf{R} \overset{\text{SVD}}{=} \mathbf{U}\,\mathbf{D}\,\mathbf{V}^T$$

$$\tilde{\mathbf{R}} = \mathbf{U}\,\mathbf{V}^T$$

Remember: an orthogonal matrix **S** decompose into

$$\mathbf{S} \overset{\text{SVD}}{=} \mathbf{U}\,\mathbf{I}\,\mathbf{V}^T$$

In general R won't be orthogonal, so

$$\mathbf{R} \overset{\text{SVD}}{=} \mathbf{U}\,\mathbf{D}\,\mathbf{V}^T \quad with \quad \mathbf{D} \neq \mathbf{I}$$

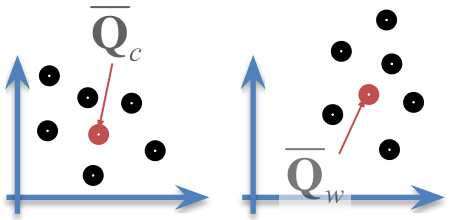
aka Orthogonal Procrustes problem

PnP

The translation $\mathbf{t}$ is straightforward…

$\begin{cases} \overline{\mathbf{Q}}_c \\ \overline{\mathbf{Q}}_w \end{cases}$ The center of mass in of the set in the camera coordinates

The center of mass in of the set in the world coordinates



$$\begin{cases} \mathbf{Q}'_c = \mathbf{Q}_c - \overline{\mathbf{Q}}_c \\ \mathbf{Q}'_w = \mathbf{Q}_w - \overline{\mathbf{Q}}_w \end{cases} \quad (1)$$

the points expressed in their barycentric coordinate systems

We have estimated $\widetilde{\mathbf{R}}$ so that $\mathbf{Q}'_c = \widetilde{\mathbf{R}}\,\mathbf{Q}'_w$ (2)

Substitute (2) in (1):

$$\mathbf{Q}_c - \overline{\mathbf{Q}}_c = \widetilde{\mathbf{R}}\left(\mathbf{Q}_w - \overline{\mathbf{Q}}_w\right)$$

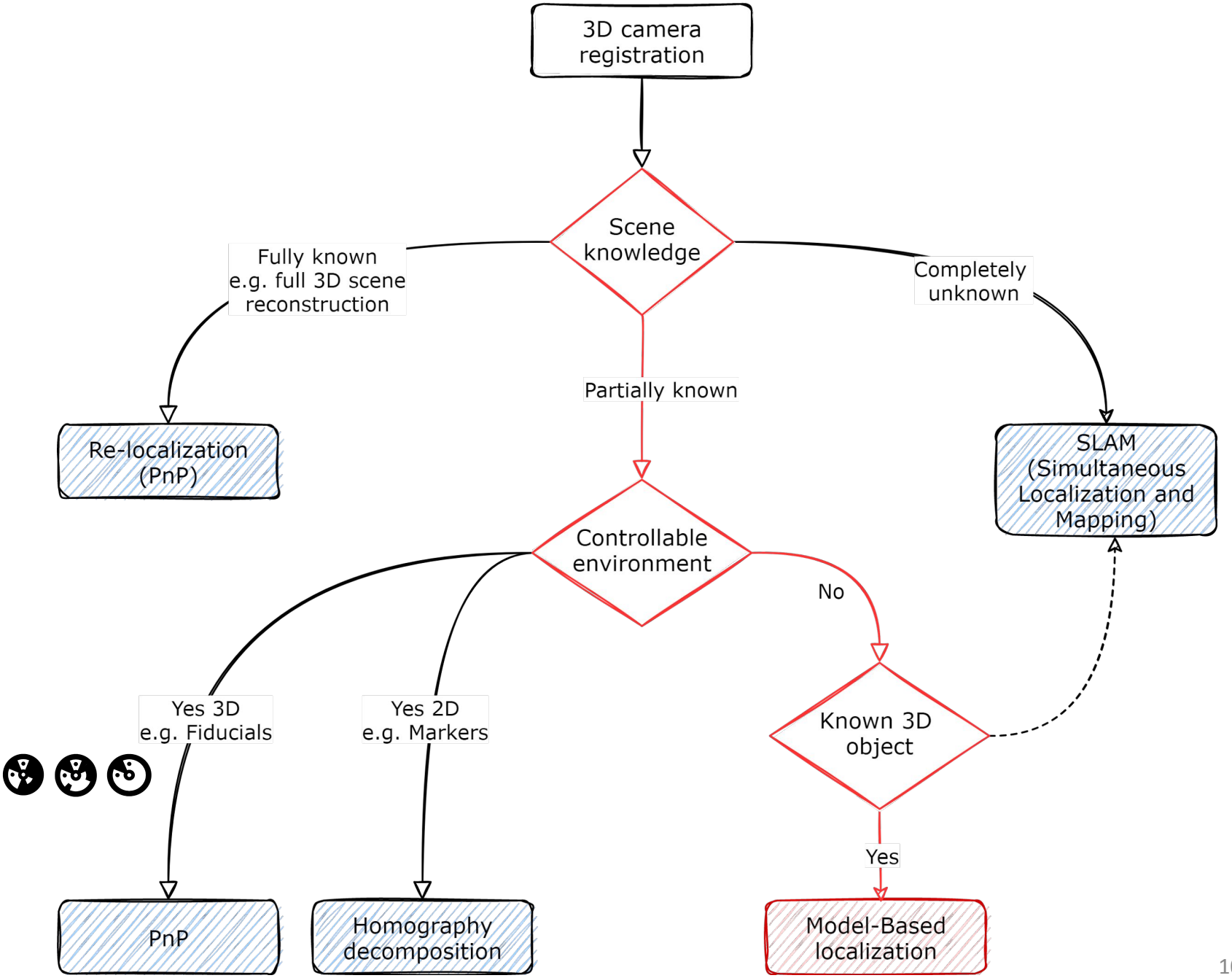
$$\mathbf{Q}_c = \widetilde{\mathbf{R}}\,\mathbf{Q}_w - \widetilde{\mathbf{R}}\overline{\mathbf{Q}}_w + \overline{\mathbf{Q}}_c$$

$$\mathbf{t} = -\widetilde{\mathbf{R}}\overline{\mathbf{Q}}_w + \overline{\mathbf{Q}}_c$$

Pose estimation with P3P

- Given N 2D-3D correspondences
- Use RANSAC framework to estimate the points in camera frame
  - Eliminate unfeasible solutions
  - Disambiguate using 4^{th} point or all the others
  - Choose the one with the smallest reprojection error
- Compute the pose R,t

- OpenCV provides `solvePnPRansac()`


3D camera registration
Scene knowledge
Fully known e.g. full 3D scene reconstruction
Completely unknown
Partially known
Re-localization (PnP)
SLAM (Simultaneous Localization and Mapping)
Controllable environment
No
Yes 3D e.g. Fiducials
Yes 2D e.g. Markers
Known 3D object
Yes
PnP
Homography decomposition
Model-Based localization

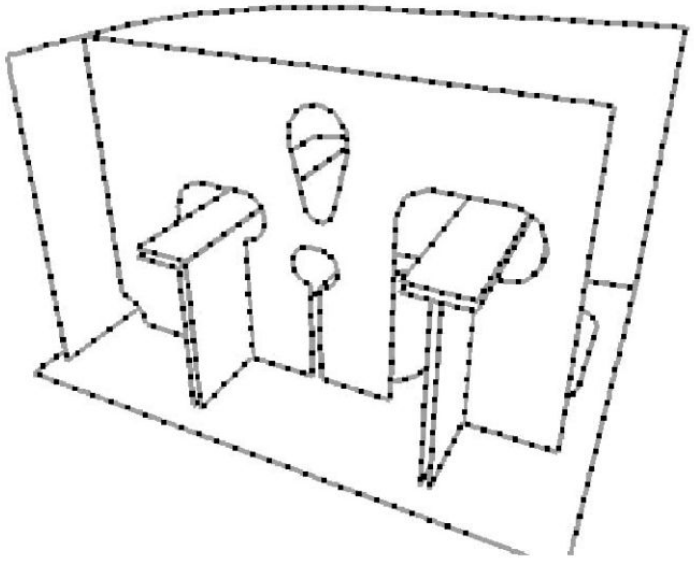
Model based tracking

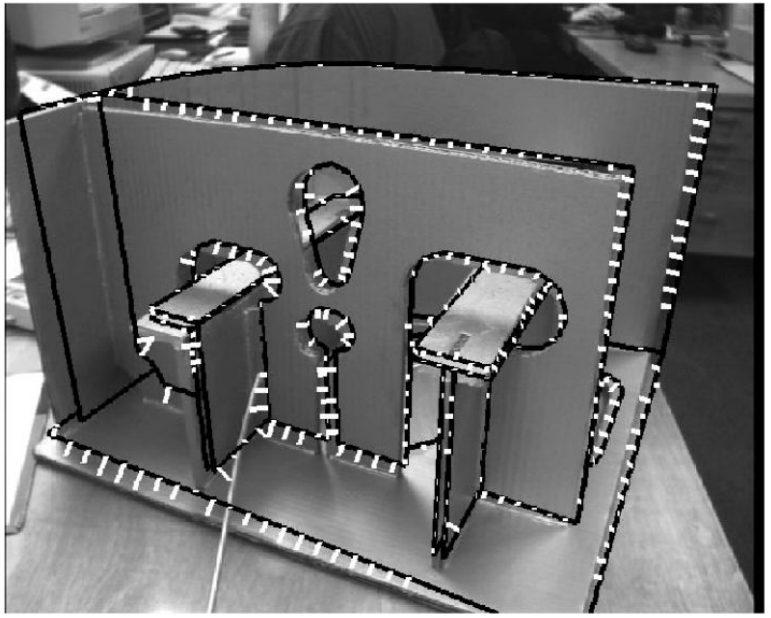
- What if we know the 3D model of an object in the scene?
  - Eg. the CAD model is available

Volkswagen MARTA project

Model based tracking – edge based

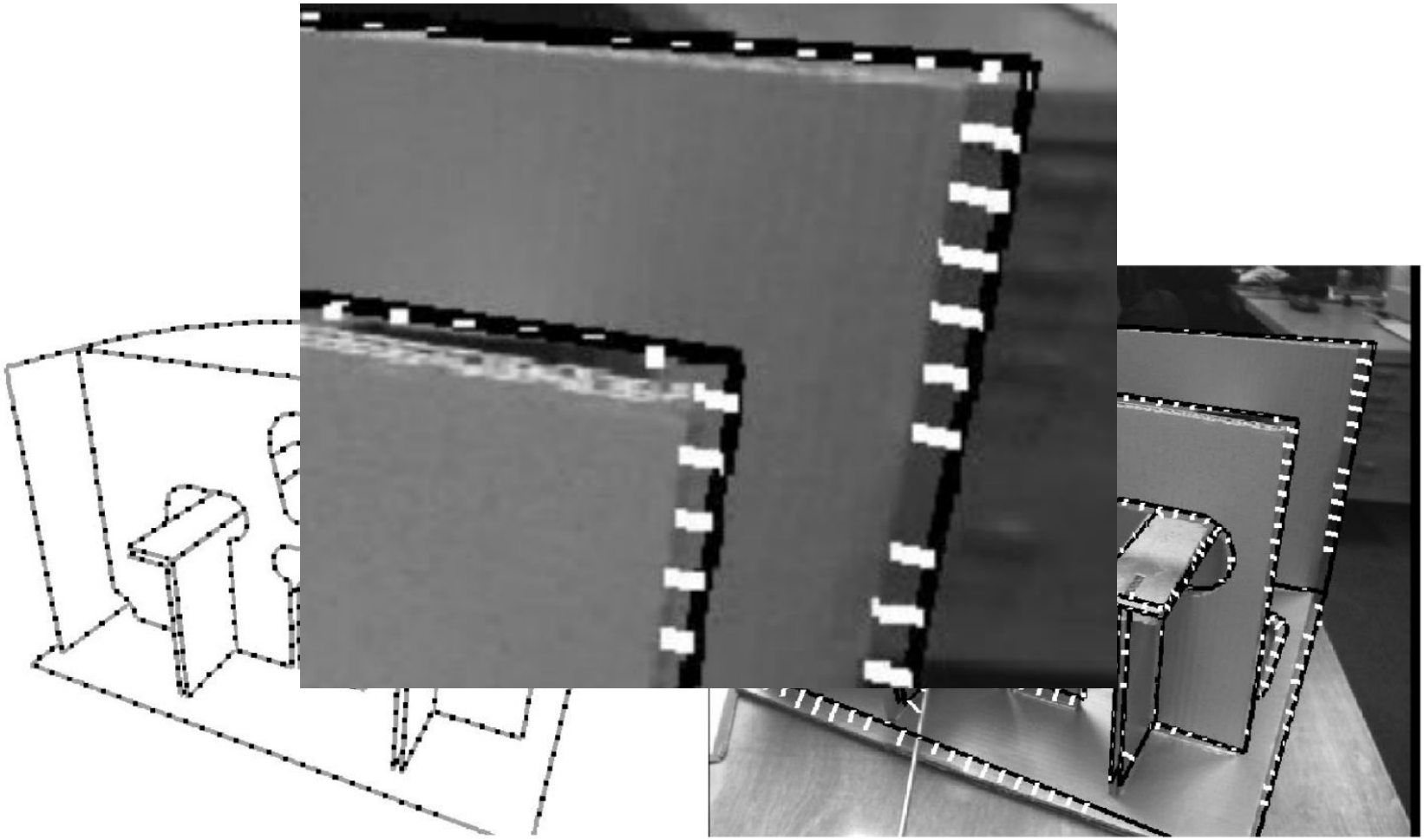
- Assuming small motions between frames
- At each frame
  - re-project sampled edge (rendering with occlusions)
  - Find the closest edge along its normal
  - Find the incremental rototranslation minimizing the distance





Real-Time Visual Tracking of Complex Structures

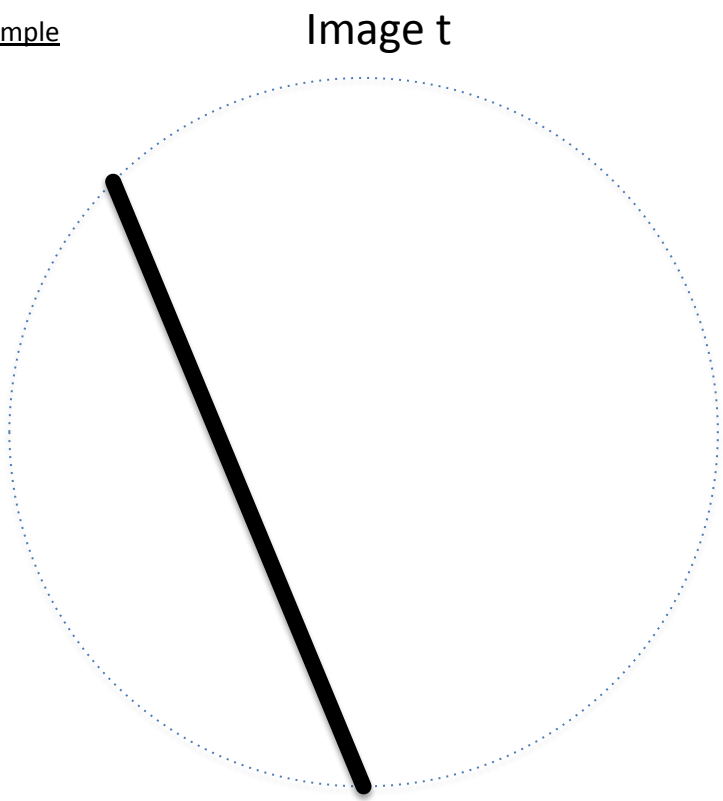
Model based tracking



Model based tracking

- Edge tracking: the aperture problem
  - Remember optical flow tracking?
- We can only estimate the movement along the edge normal

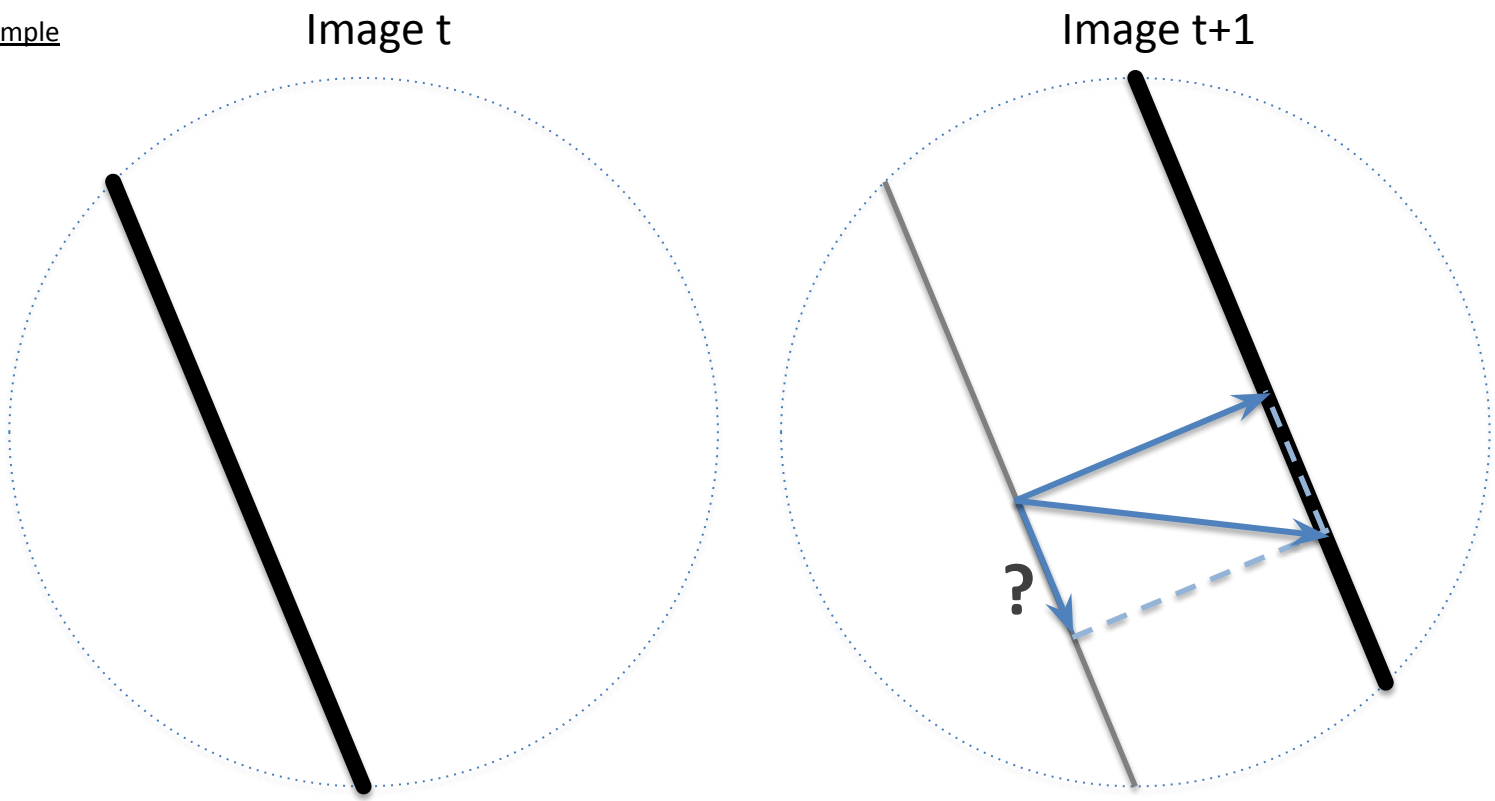
Animated example



Model based tracking

- Edge tracking: the aperture problem
  - Remember optical flow tracking?
- We can only estimate the movement along the edge normal

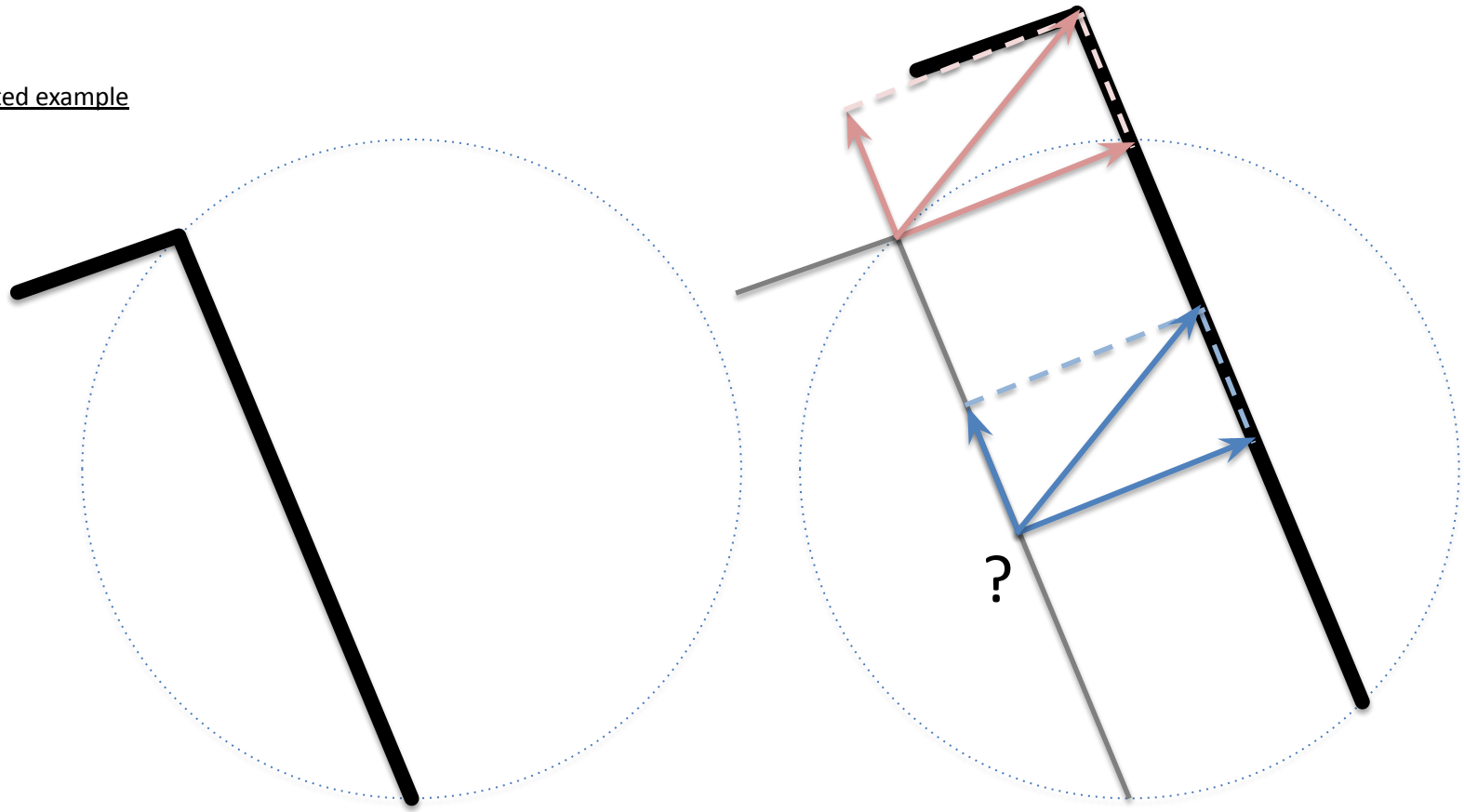
Animated example



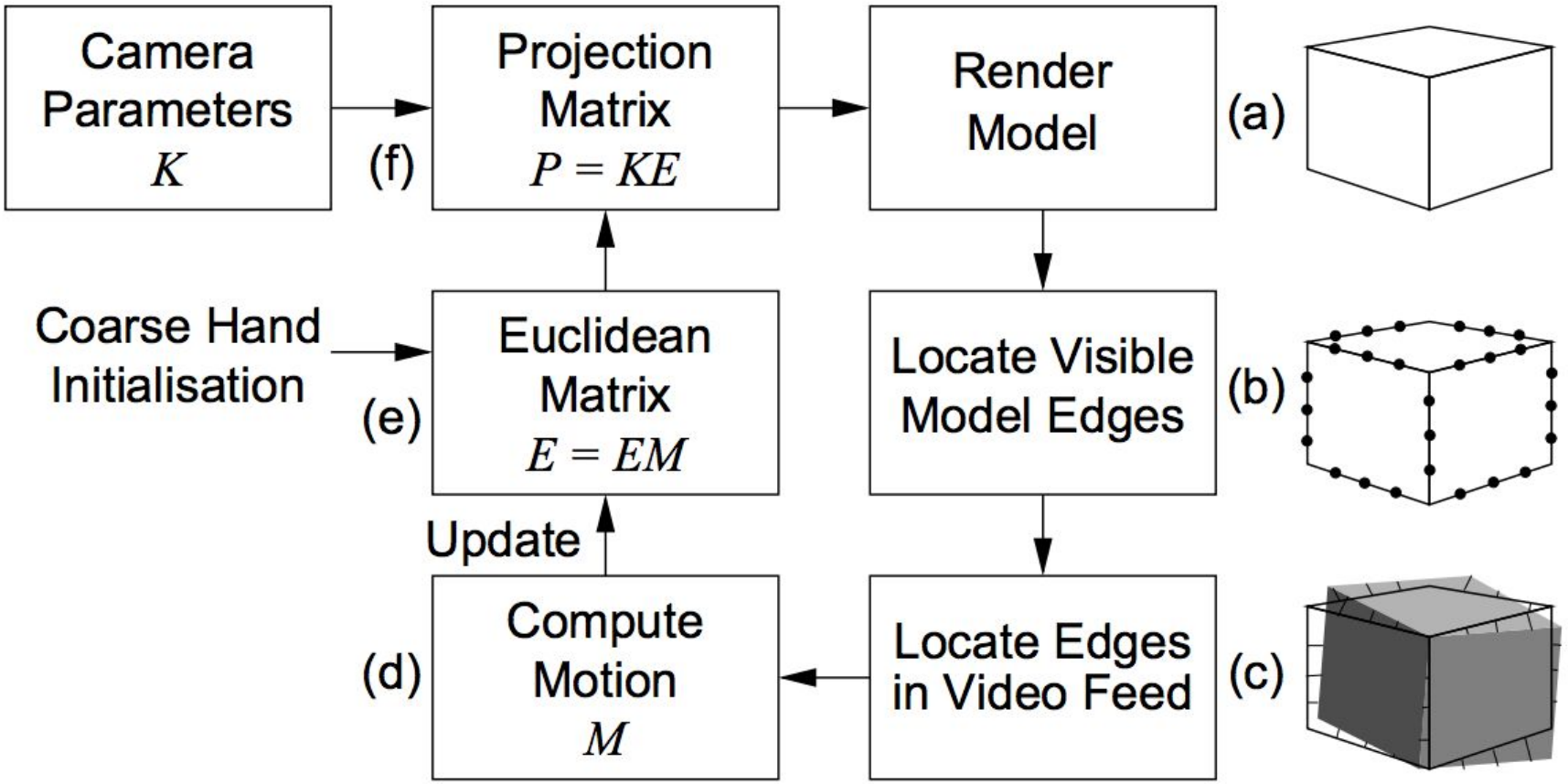
Model based tracking

- Edge tracking: the aperture problem
  - Remember optical flow tracking?
- We can only estimate the movement along the normal

Animated example



Model based tracking



To resume

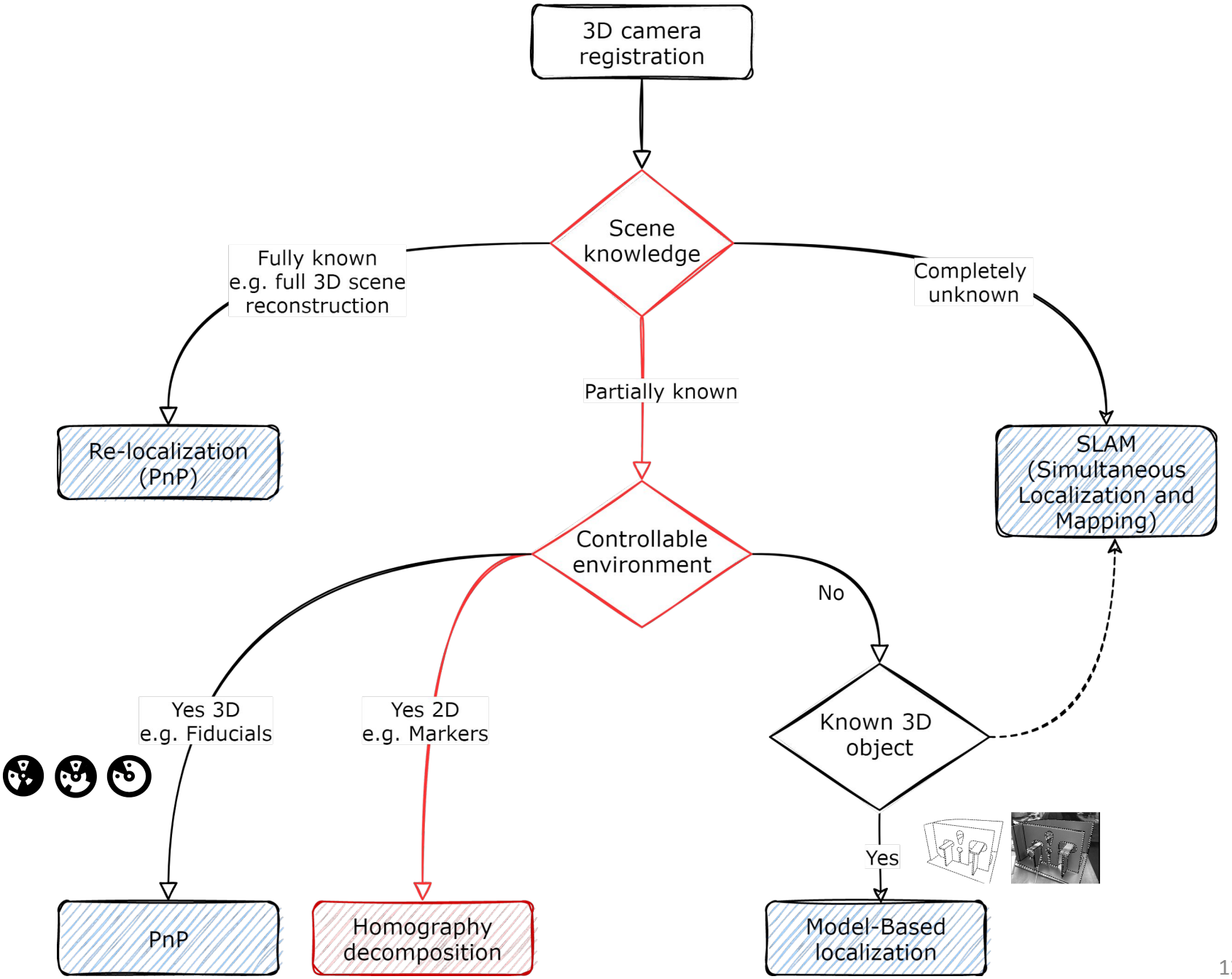
- Exploit the known 3D information
- Fiducial points placed in the scene
  - Detect the point
  - Solve the resection (PnP) problem
- 3D model of an object to track
  - Use its CAD model
    - To track edges
    - To track its point cloud

Today Topics

- The registration problem
- Optical visual tracking for AR
  - Background – camera model and calibration
  - Pose estimation using fiducials
    - Model Based tracking
  - Pose estimation using markers

The registration problem

- Assumptions about the scene (What do we know?)
  - Some 3D references in the scene (fiducials)
  - The 3D model of an object to track (model-based tracking)
  - Some 2D reference in the scene (markers)
  - Nothing? Use the natural features (SLAM)

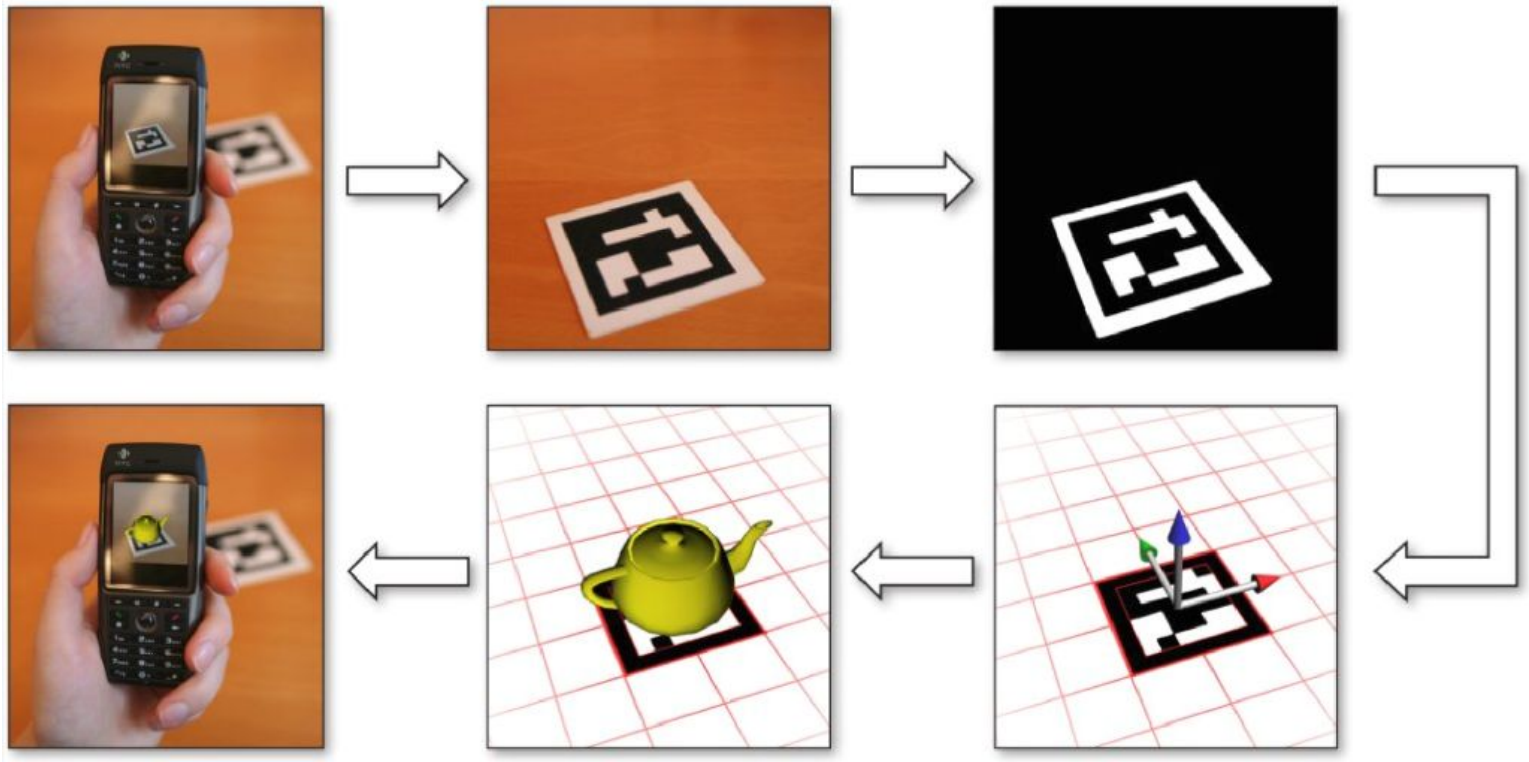

3D camera registration
Scene knowledge
Fully known e.g. full 3D scene reconstruction
Completely unknown
Partially known
Re-localization (PnP)
SLAM (Simultaneous Localization and Mapping)
Controllable environment
No
Yes 3D e.g. Fiducials
Yes 2D e.g. Markers
Known 3D object
Yes
PnP
Homography decomposition
Model-Based localization

Marker-based tracking

- Easiest way to do augmented reality
  - Use the marker(s) as placeholder(s) for the virtual object(s)
  - Standard image processing and computer vision techniques
  - Suitable even for common mobile phones
- Technique done for more than 10 years
- Several open source tools available
- Drawbacks
  - Set up the scene with markers (not always possible)
  - Digitally remove the markers from the image

Marker-based tracking

- General pipeline



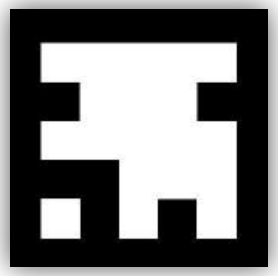
Marker tracking

1. Image acquisition
2. Preprocessing
   - Undistortion
   - Low level feature extraction
3. Test candidates
   - Fast rejection of non-markers
   - Fast acceptance of markers
4. Marker identification
   - Decoding (data markers)
   - Template matching (template matching)
5. Pose estimation
   - Estimation refinement

Marker tracking

Image acquisition

- Conversion to greylevel image
- Working with color is hard!
  - Automatic white balance
  - Color change according to light condition and scene
- Differences in luminance (brightness) are easier to detect
- Therefore markers usually are:
  - Black and white: high contrast ease detection
  - Squared: 4 points are enough to estimate the pose (see later)

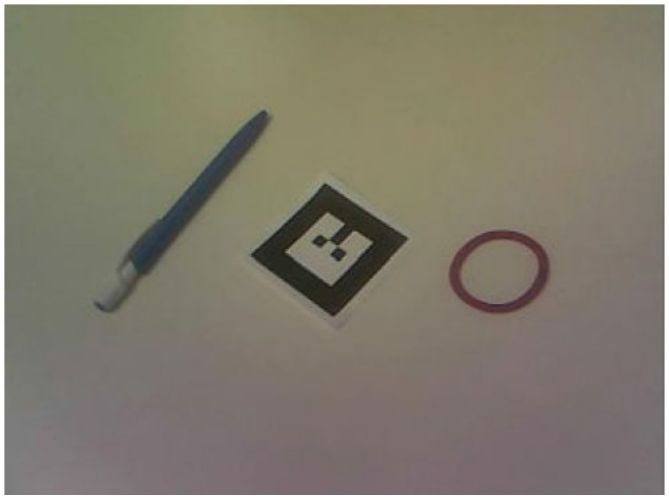


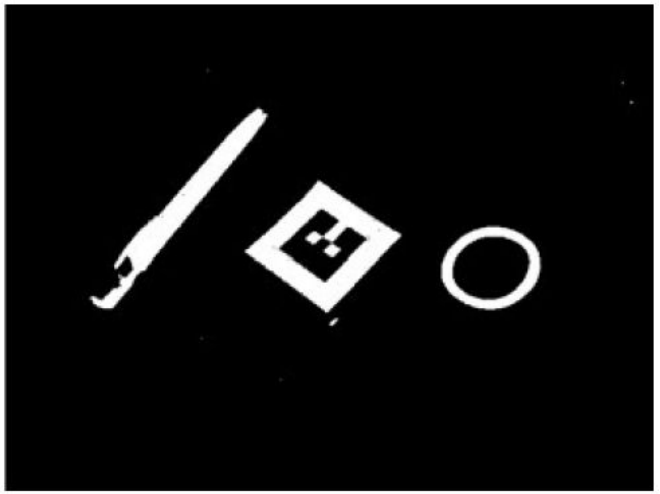
Marker tracking

Preprocessing

- Undistortion
  - Remove the optical distortion (calibrated camera)
  - Look-up table to speed-up
  - Global undistortion Vs Local undistortion (faster and optimized)
- Thresholding

$$I_{thresh} = I < thresh$$





Marker tracking

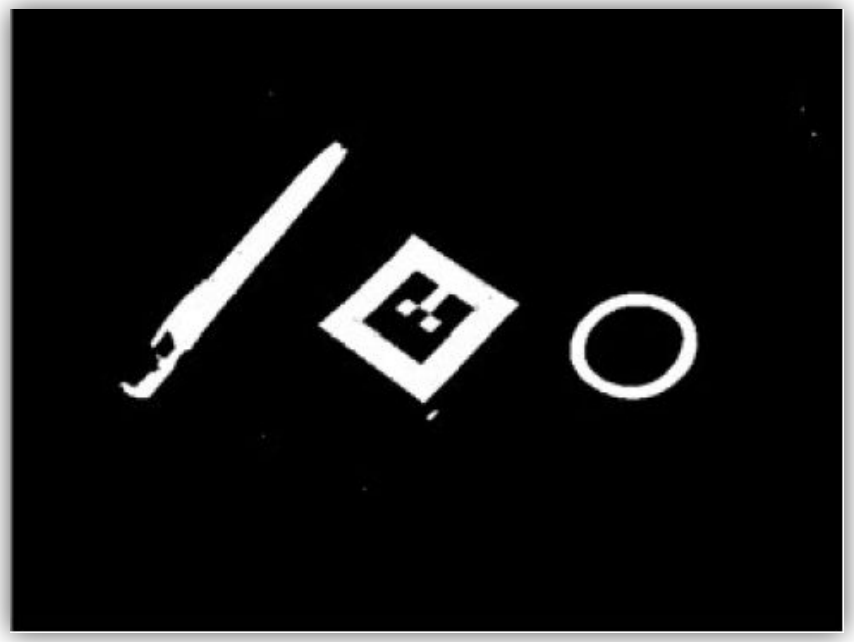
Preprocessing

- Adaptive thresholding
  - Different threshold values for each pixel to account for light variations
  - Smaller image regions are more likely to have uniform illumination
  - Histogram-based
    - For each region compute the histogram and select the threshold
    - Slow
  - Statistic-based
    - For each pixel compute some statistics for the neighbor pixels
    - mean, median, (min+max)/2 …

Marker tracking

Preprocessing

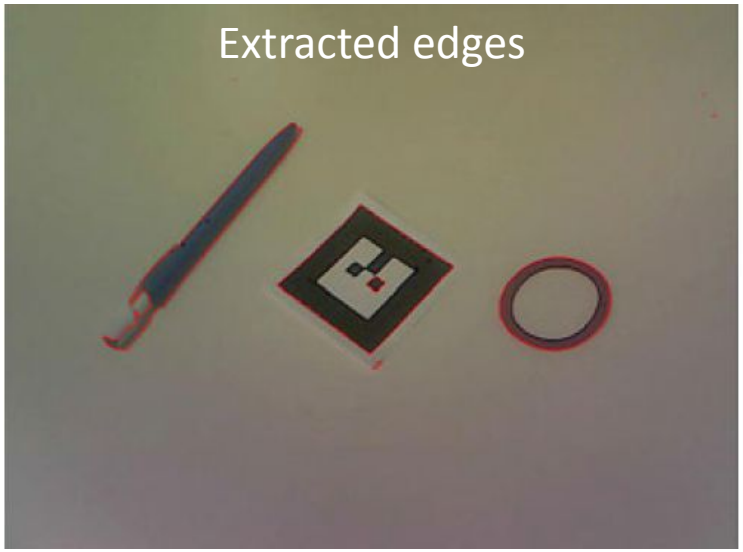
- Blob analysis
  - Labeling connected regions
  - Compute statistics for each blob (size, perimeter, area)
  - Used later for fast rejection/acceptance

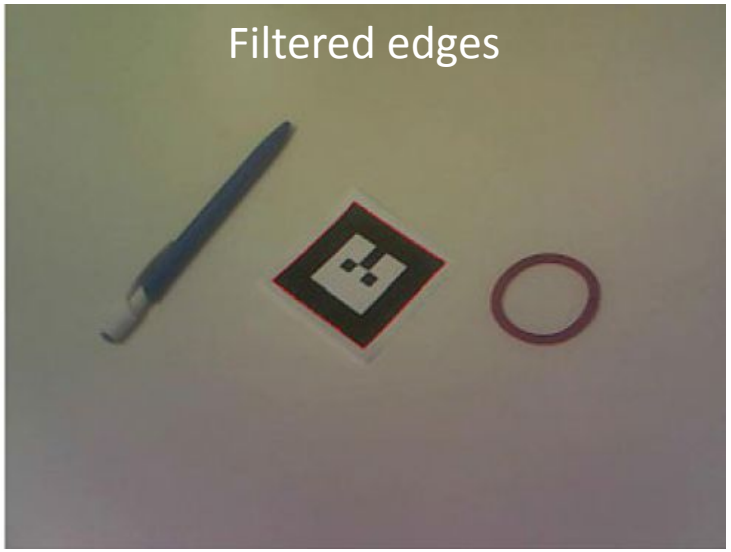


Marker tracking

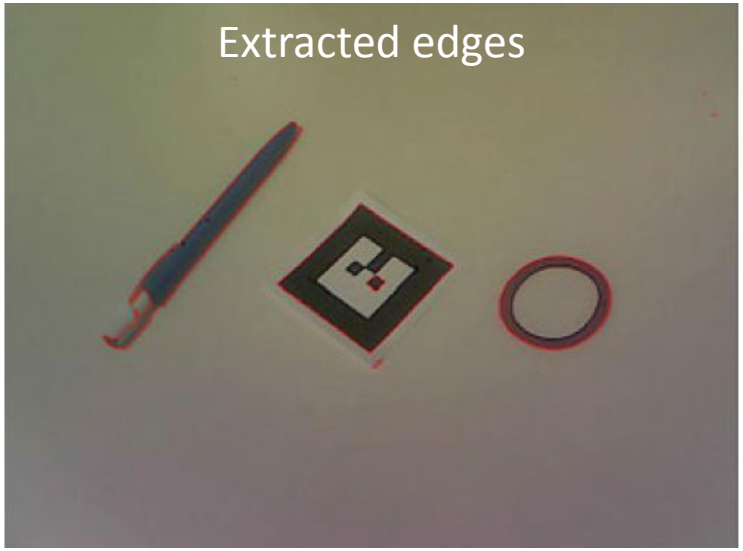
Preprocessing

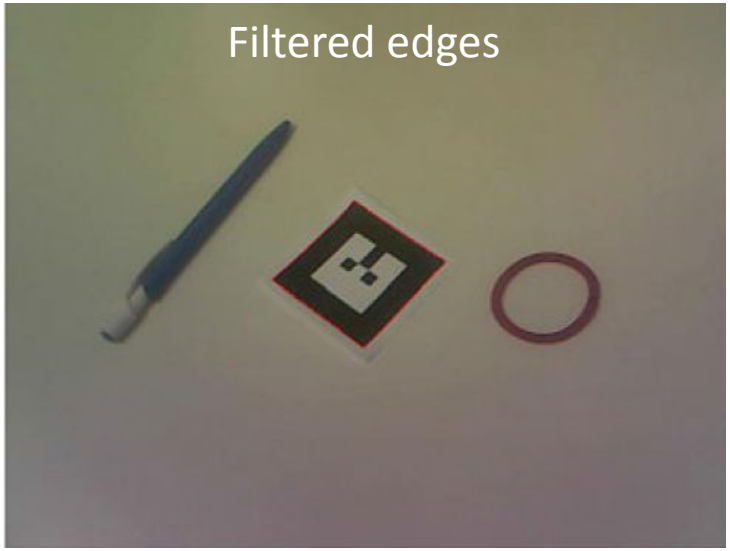
- Edge detection
  - Detect discontinuity/border pixel in each blob
  - Apply quadrilateral test (4 borders, 4 corners)
  - [Local undistortion]
  - Line fitting on borders
  - Refine corner location to sub-pixel accuracy

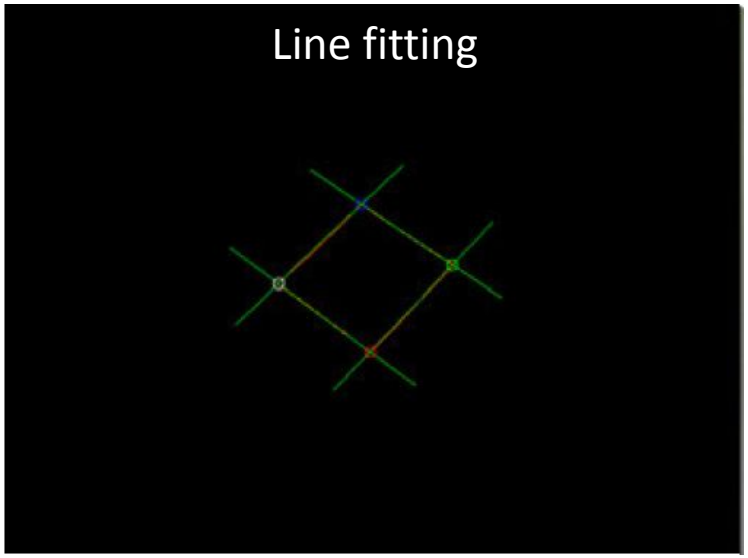


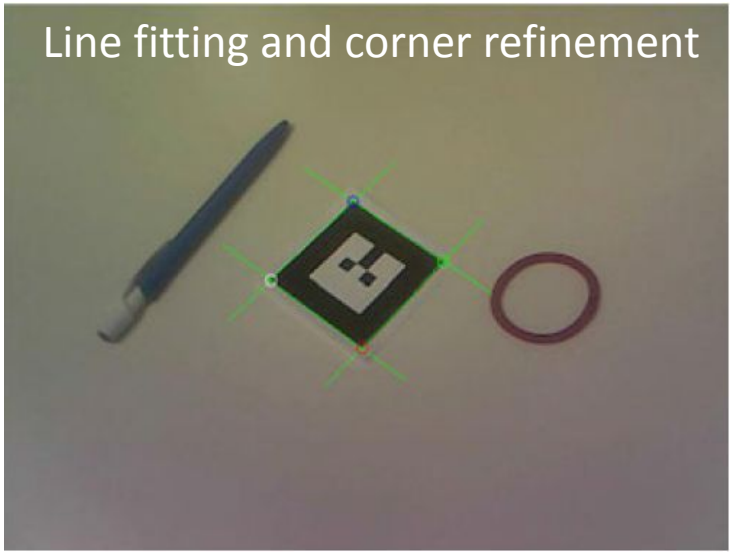


Marker tracking









Marker tracking

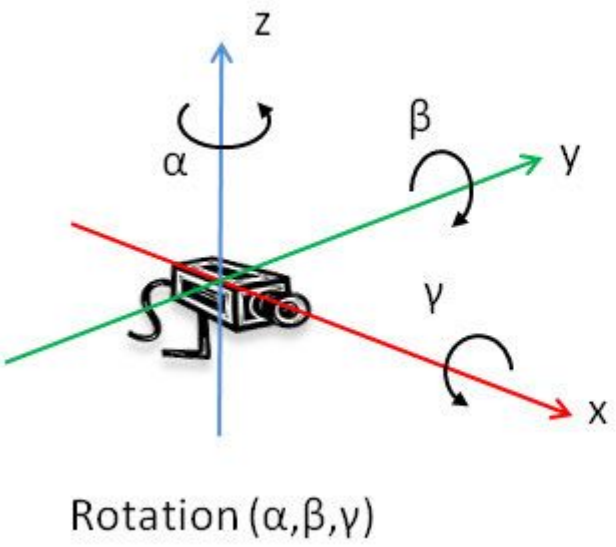
Preprocessing

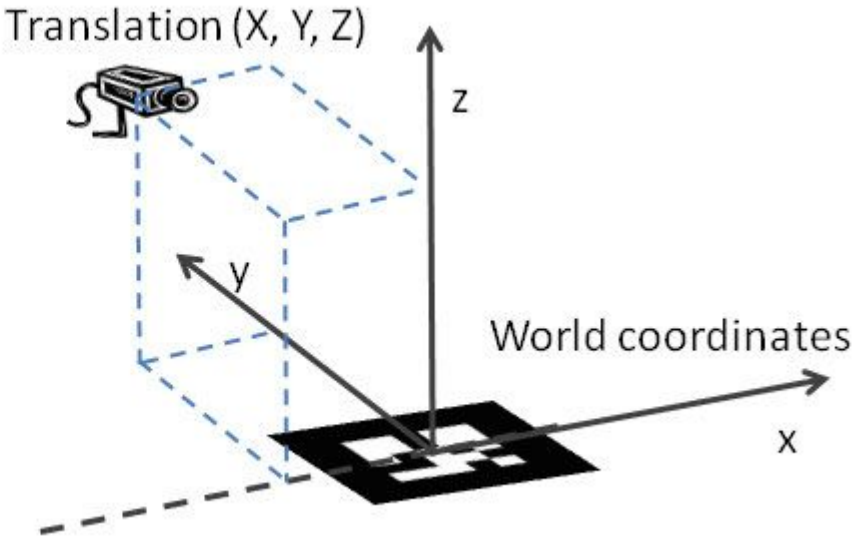
- Corner refinement
  - Critical for high accuracy tracking
  - 4 points are the minimum
  - Hence they must be as accurate as possible
- Harris corner detection
- Remove optical distortion
- Estimate can be affected by
  - Motion blur
  - Image noise
  - Pixel quantization errors…

Marker tracking

Pose estimation

- 6DOF
  - The orientation (3 angles)
  - The position (3 coordinates)
- 4 points enough in the calibrated case

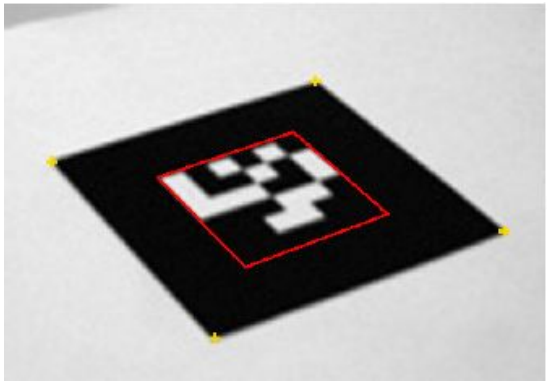


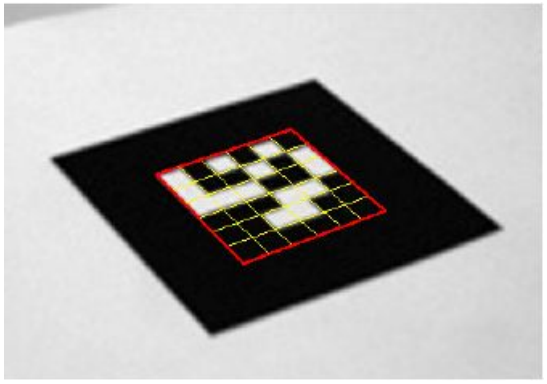


Marker tracking

Marker identification

- Homography from 4 corners to eliminate perspective distortion
- Check the pattern
  - Decoding (data markers)
  - Template matching (template markers)





Planar homography

$$\mathbf{q} \sim \mathtt{K} \begin{bmatrix} \mathtt{R} & \mathbf{t} \end{bmatrix} \hat{\mathbf{Q}} = \mathtt{K} \begin{bmatrix} \mathtt{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} \mathbf{Q} \\ 1 \end{bmatrix}$$

projection equation of a point **Q** of the marker

Since the reference system R, t is arbitrary we can choose one that is aligned with one of the corners and with the marker contained in the plane z=0. The coordinates of the point on the marker then can be written as

$$\hat{\mathbf{Q}} = \begin{bmatrix} Q_x \\ Q_y \\ 0 \\ 1 \end{bmatrix}$$

So the projection equation for points on the marker becomes

$$\mathbf{q} \sim \mathtt{K} \begin{bmatrix} \mathtt{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} Q_x \\ Q_y \\ 0 \\ 1 \end{bmatrix} = \mathtt{K} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \begin{bmatrix} Q_x \\ Q_y \\ 1 \end{bmatrix}$$

with $\mathbf{r}_1$ and $\mathbf{r}_2$ the first and second column of the matrix **R**, respectively

$$\mathbf{q} \sim \mathtt{H}_{3\times3} \begin{bmatrix} Q_x \\ Q_y \\ 1 \end{bmatrix}$$

Homography matrix **H**

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

Marker tracking

Pose estimation

- Using Homography
  - Estimate the homography from 4 corners
  - Decompose the homography to get R and t

$$q \sim \mathbf{H}Q = \mathbf{K}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix}Q$$

$$\mathbf{K}^{-1}\mathbf{H} = \mathbf{K}^{-1}\begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix} = \lambda\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix}$$

To be implemented during the TP!

Marker tracking

Pose estimation

- Using Homography
  - Estimate the homography from 4 corners
  - Decompose the homography to get R and t

$$q \sim \mathbf{H}Q = \mathbf{K}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix}Q$$

$$\mathbf{K}^{-1}\mathbf{H} = \mathbf{K}^{-1}\begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix} = \lambda\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix}$$

$$\begin{cases} \mathbf{r}_1 = \lambda\mathbf{K}^{-1}h_1 \\ \mathbf{r}_2 = \lambda\mathbf{K}^{-1}h_2 \\ \mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2 \\ \mathbf{t} = \lambda\mathbf{K}^{-1}h_3 \end{cases} \quad \text{with} \quad \lambda = \frac{1}{\left\|\mathbf{K}^{-1}h_1\right\|} = \frac{1}{\left\|\mathbf{K}^{-1}h_2\right\|}$$

To be implemented during the TP!

Marker tracking

Pose estimation

- Using Homography
  - Estimate the homography from 4 corners
  - Decompose the homography to get R and t

$$q \sim \mathbf{H}Q = \mathbf{K}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix}Q$$

$$\mathbf{K}^{-1}\mathbf{H} = \mathbf{K}^{-1}\begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix} = \lambda\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix}$$

$$\begin{cases} \mathbf{r}_1 = \lambda\mathbf{K}^{-1}h_1 \\ \mathbf{r}_2 = \lambda\mathbf{K}^{-1}h_2 \\ \mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2 \\ \mathbf{t} = \lambda\mathbf{K}^{-1}h_3 \end{cases} \quad \text{with} \quad \lambda = \frac{1}{\left\|\mathbf{K}^{-1}h_1\right\|} = \frac{1}{\left\|\mathbf{K}^{-1}h_2\right\|}$$

To be implemented during the TP!

The obtained
R = [r1 r2 r3]
is in general not orthogonal due to noise:
Use SVD to find the "closest" one , ie:

$$\mathbf{R} = \mathbf{U}\mathbf{S}\mathbf{V}^T$$
$$\tilde{\mathbf{R}} = \mathbf{U}\mathbf{V}^T$$

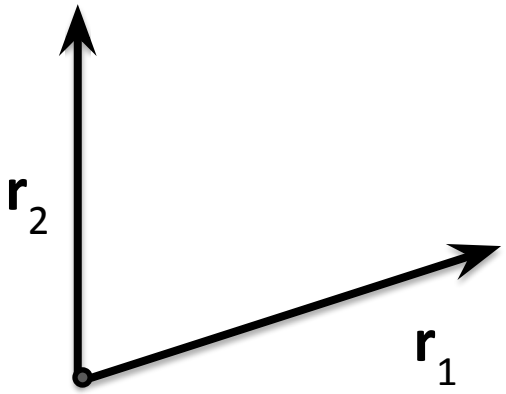
Marker tracking

Pose estimation

- Using Homography
  - Estimate the homography from 4 corners
  - Decompose the homography to get R and t

Another approach (general case)

$$\lambda = \frac{1}{\sqrt{\|\mathbf{h}_1\|\|\mathbf{h}_2\|}} \quad \mathbf{r}_1 = \lambda \mathbf{h}_1 \quad \mathbf{r}_2 = \lambda \mathbf{h}_2 \quad \mathbf{t} = \lambda \mathbf{h}_3$$



Marker tracking

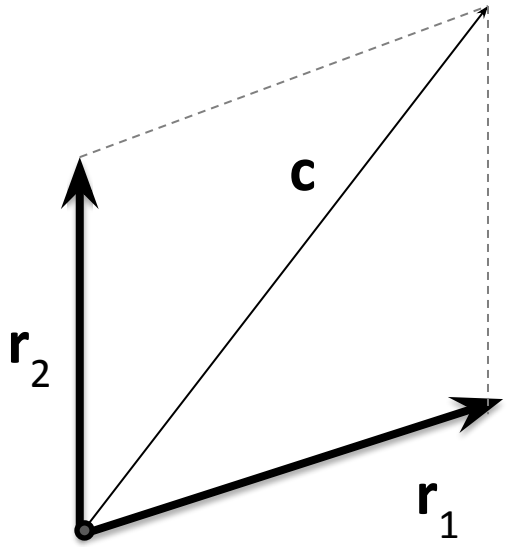
Pose estimation

- Using Homography
  - Estimate the homography from 4 corners
  - Decompose the homography to get R and t

Another approach (general case)

$$\lambda = \frac{1}{\sqrt{\|\mathbf{h}_1\|\|\mathbf{h}_2\|}} \quad \mathbf{r}_1 = \lambda \mathbf{h}_1 \quad \mathbf{r}_2 = \lambda \mathbf{h}_2 \quad \mathbf{t} = \lambda \mathbf{h}_3$$

$$\mathbf{c} = \mathbf{r}_1 + \mathbf{r}_2$$



Marker tracking

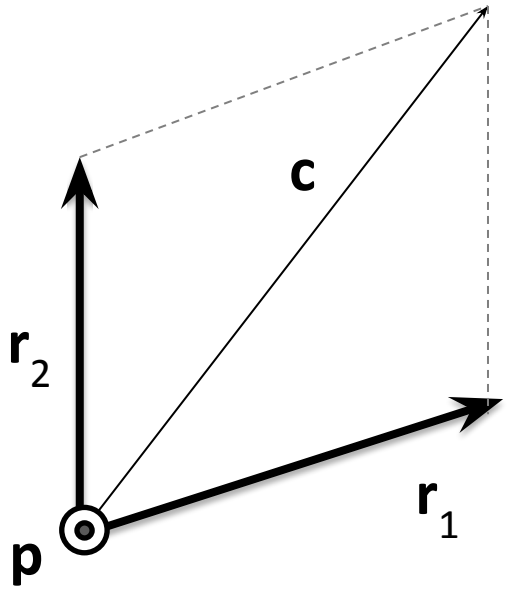
Pose estimation

- Using Homography
  - Estimate the homography from 4 corners
  - Decompose the homography to get R and t

Another approach (general case)

$$\lambda = \frac{1}{\sqrt{\|\mathbf{h}_1\|\|\mathbf{h}_2\|}} \quad \mathbf{r}_1 = \lambda\mathbf{h}_1 \quad \mathbf{r}_2 = \lambda\mathbf{h}_2 \quad \mathbf{t} = \lambda\mathbf{h}_3$$

$$\mathbf{c} = \mathbf{r}_1 + \mathbf{r}_2 \quad \mathbf{p} = \mathbf{r}_1 \times \mathbf{r}_2$$



p is “pointing outwards perpendicularly to this slide”

Marker tracking

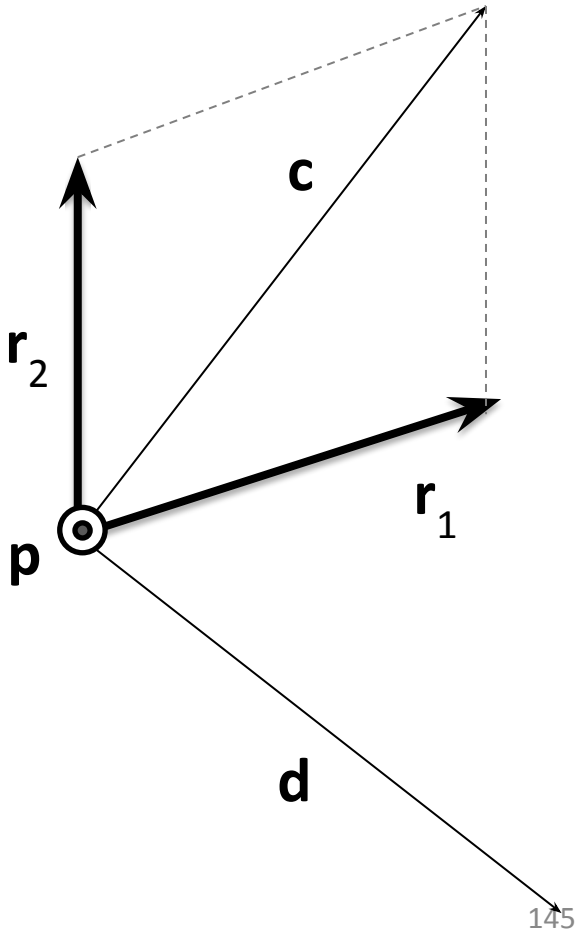
Pose estimation

- Using Homography
  - Estimate the homography from 4 corners
  - Decompose the homography to get R and t

Another approach (general case)

$$\lambda = \frac{1}{\sqrt{\|\mathbf{h}_1\|\|\mathbf{h}_2\|}} \quad \mathbf{r}_1 = \lambda \mathbf{h}_1 \quad \mathbf{r}_2 = \lambda \mathbf{h}_2 \quad \mathbf{t} = \lambda \mathbf{h}_3$$

$$\mathbf{c} = \mathbf{r}_1 + \mathbf{r}_2 \quad \mathbf{p} = \mathbf{r}_1 \times \mathbf{r}_2 \quad \mathbf{d} = \mathbf{c} \times \mathbf{p}$$



Marker tracking

Pose estimation

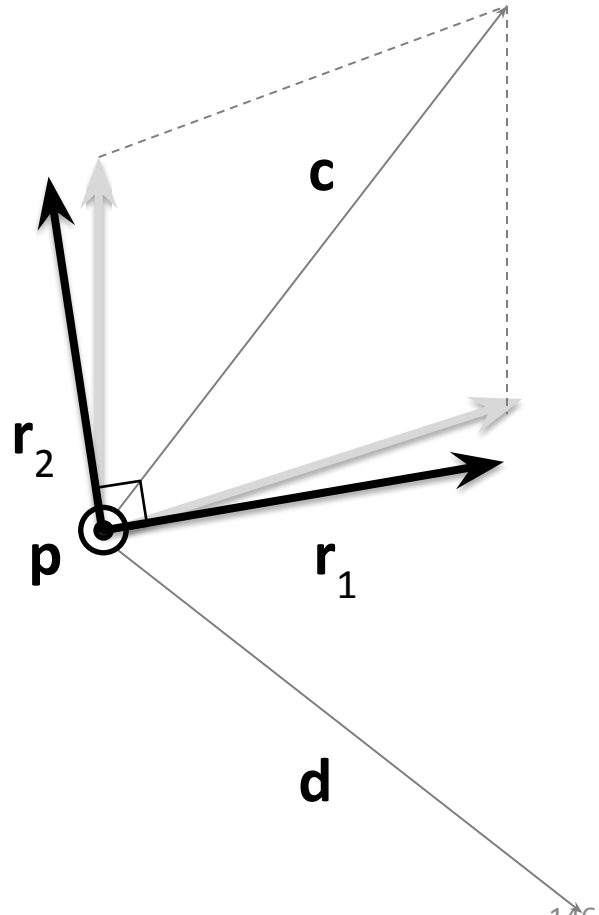
- Using Homography
  - Estimate the homography from 4 corners
  - Decompose the homography to get R and t

Another approach (general case)

$$\lambda = \frac{1}{\sqrt{\|\mathbf{h}_1\|\|\mathbf{h}_2\|}} \quad \mathbf{r}_1 = \lambda\mathbf{h}_1 \quad \mathbf{r}_2 = \lambda\mathbf{h}_2 \quad \mathbf{t} = \lambda\mathbf{h}_3$$

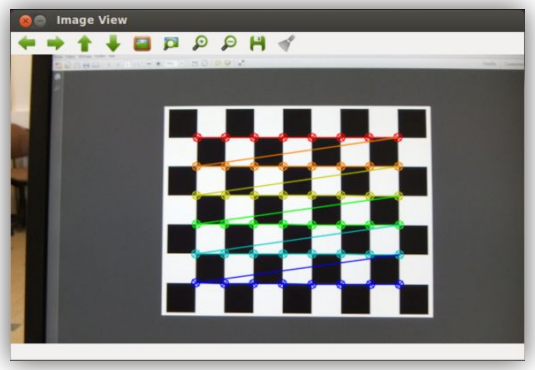
$$\mathbf{c} = \mathbf{r}_1 + \mathbf{r}_2 \quad \mathbf{p} = \mathbf{r}_1 \times \mathbf{r}_2 \quad \mathbf{d} = \mathbf{c} \times \mathbf{p}$$

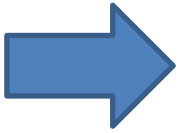
$$\mathbf{r}_1 = \frac{1}{\sqrt{2}}\left(\frac{\mathbf{c}}{\|\mathbf{c}\|} + \frac{\mathbf{d}}{\|\mathbf{d}\|}\right) \quad \mathbf{r}_2 = \frac{1}{\sqrt{2}}\left(\frac{\mathbf{c}}{\|\mathbf{c}\|} - \frac{\mathbf{d}}{\|\mathbf{d}\|}\right) \quad \mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2$$

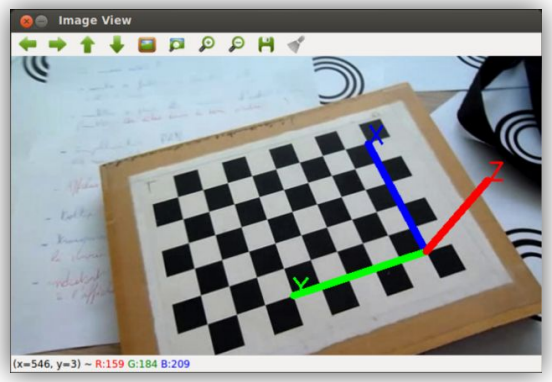


[digression] TP subject

- Idea:
  - Detect the chessboard
  - Estimate the homography
  - Decompose the homography to get the pose





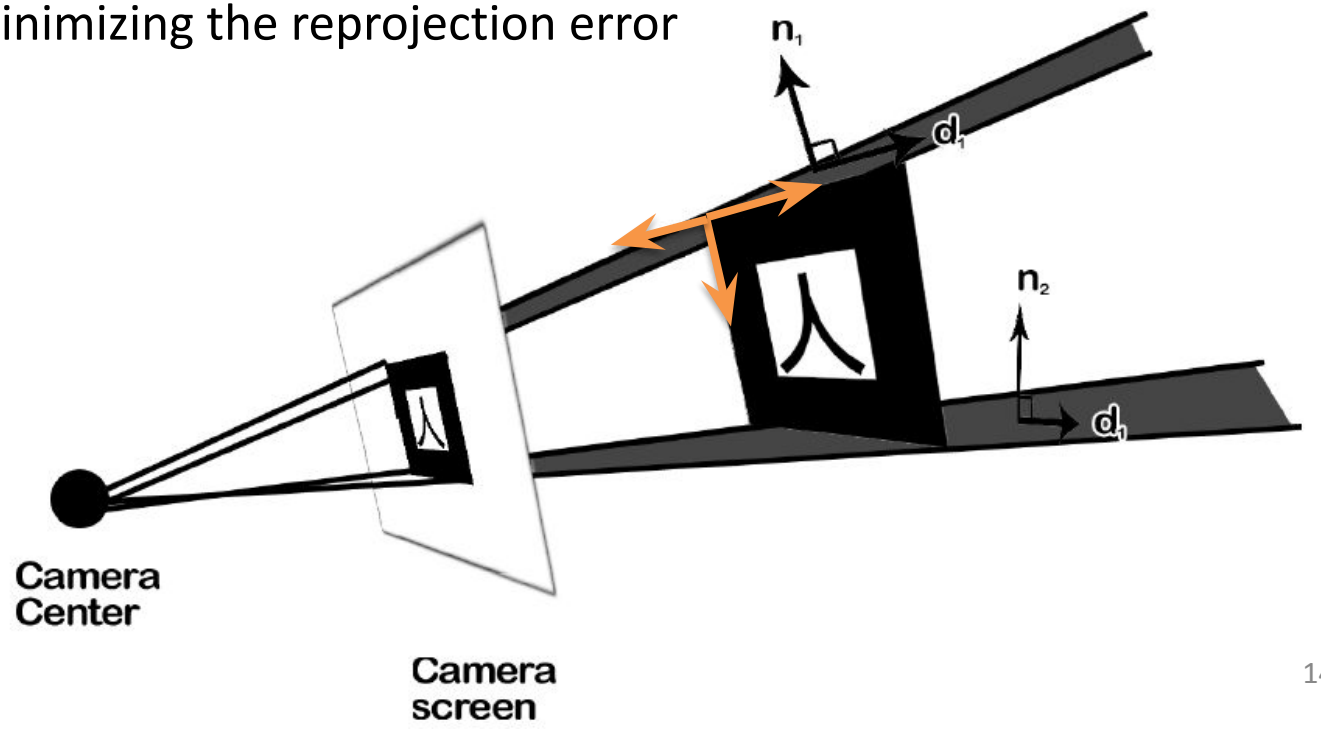


- More than 4 points available
- Estimation of H in a robust way
  - The obtained H can give a more stable decomposition
- RANSAC for estimating H (opencv `findHomography()`)

Marker tracking

Pose estimation

- The “artoolkit” method
- Main idea:
  - Use the contours lines to estimate the rotation
  - Solve for the translation t
  - Refine minimizing the reprojection error



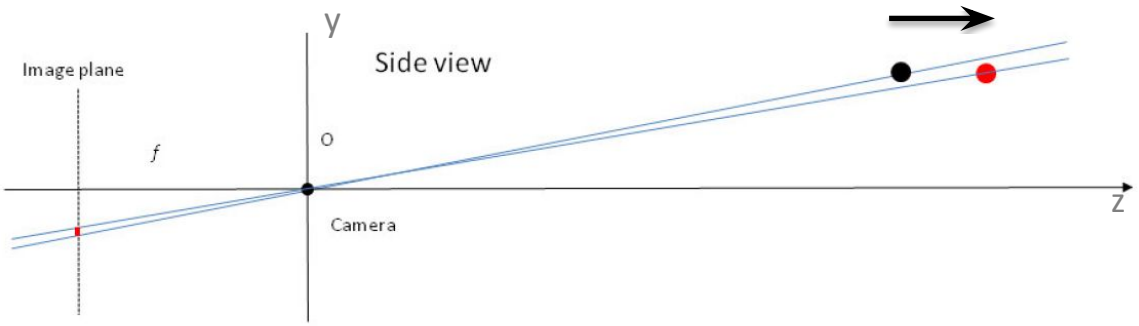
Marker tracking - jittering

https://www.youtube.com/watch?v=DqS9Dh1zrqM

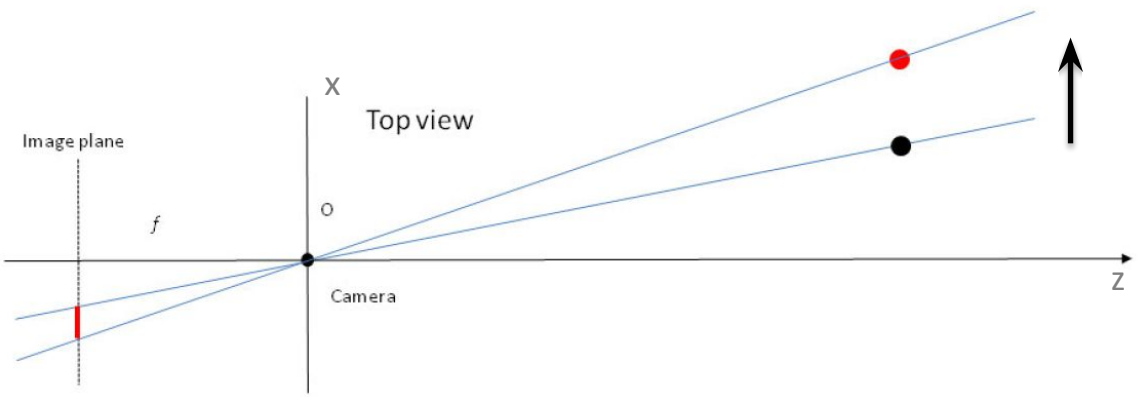
Marker tracking

Pose estimation

- In general still have some jittering
  - Small errors in detecting the 4 points may be significant



An object moving parallel to the optical axis generates a smaller movement in the image plane



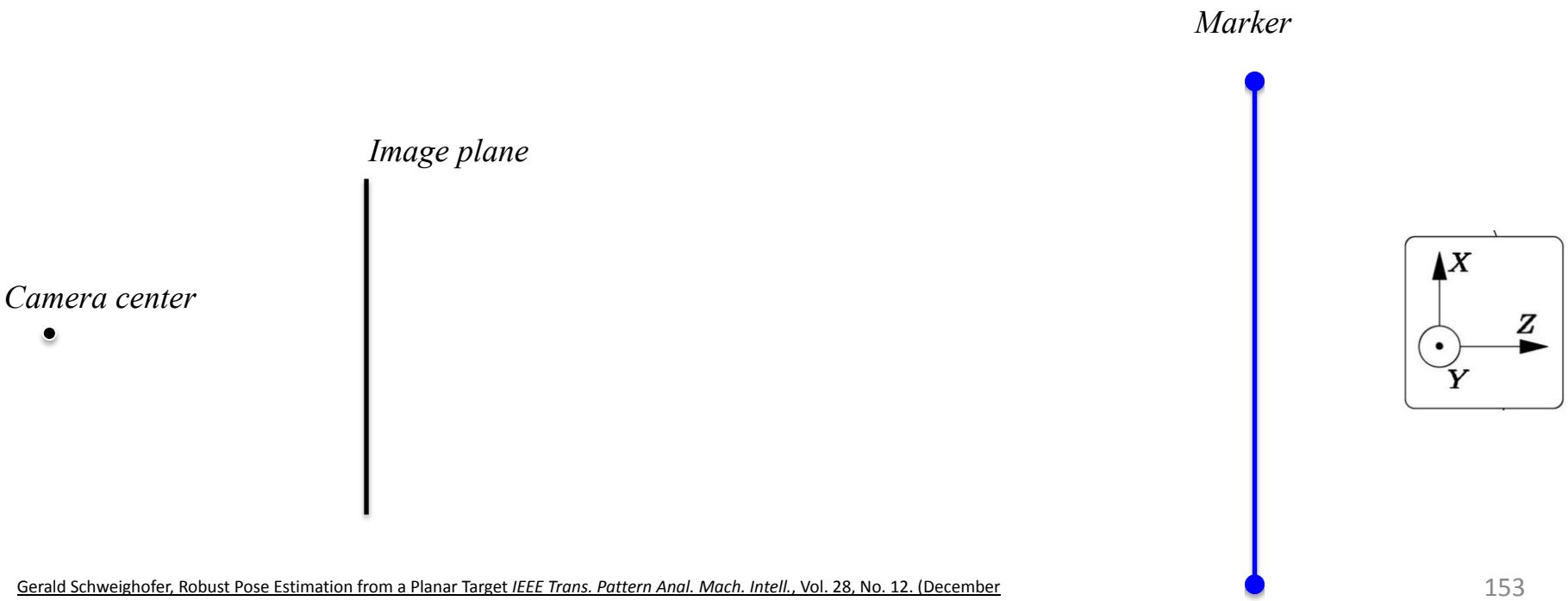
An object moving perpendicularly to the optical axis generates a larger movement in the image plane

 small detection errors on the image plane have a greater effect on the Z dimension.

Marker tracking

Pose estimation

- The Robust Planar Pose (RPP) algorithm (ArtoolkitPlus)
  - Intuitive clue: for every chosen α there may exist a second, different angle β which also leads to a local minimum of the error function

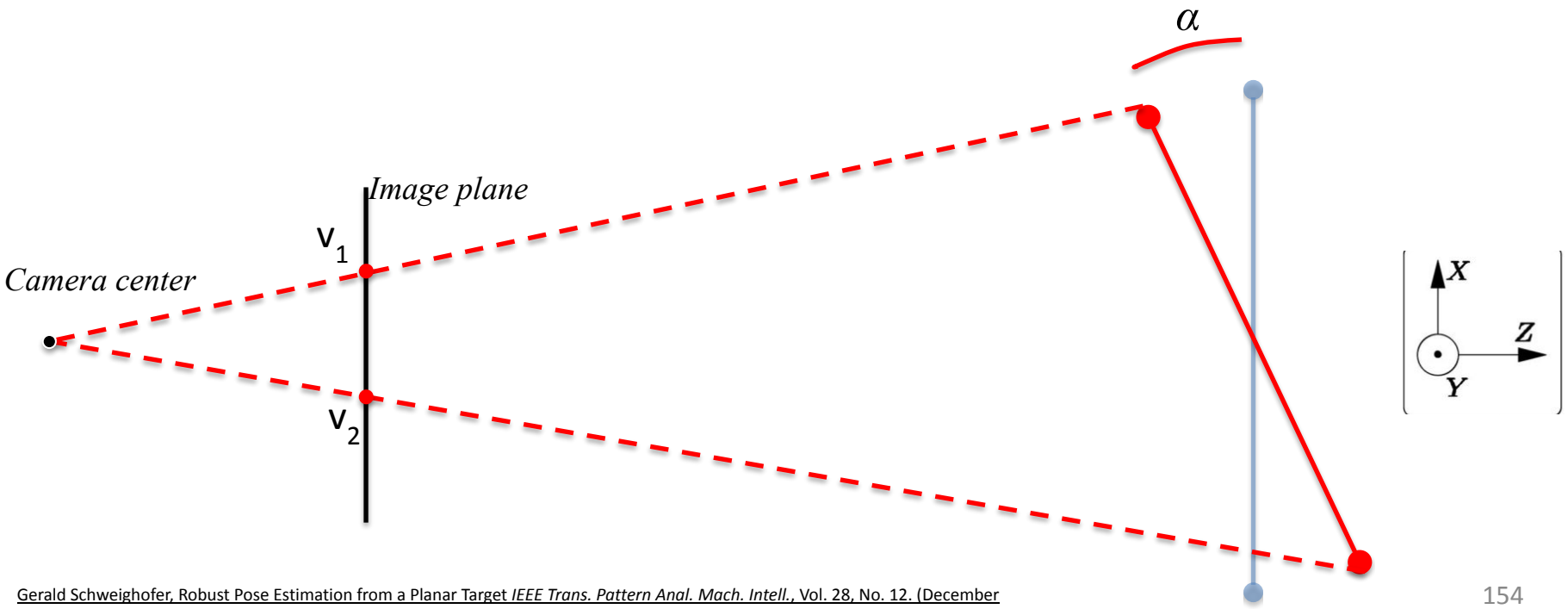


Gerald Schweighofer, Robust Pose Estimation from a Planar Target *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 28, No. 12. (December 2006), pp. 2024-2030.

Marker tracking

Pose estimation

- The Robust Planar Pose (RPP) algorithm (ArtoolkitPlus)
  - Intuitive clue: for every chosen α there may exist a second, different angle β which also leads to a local minimum of the error function

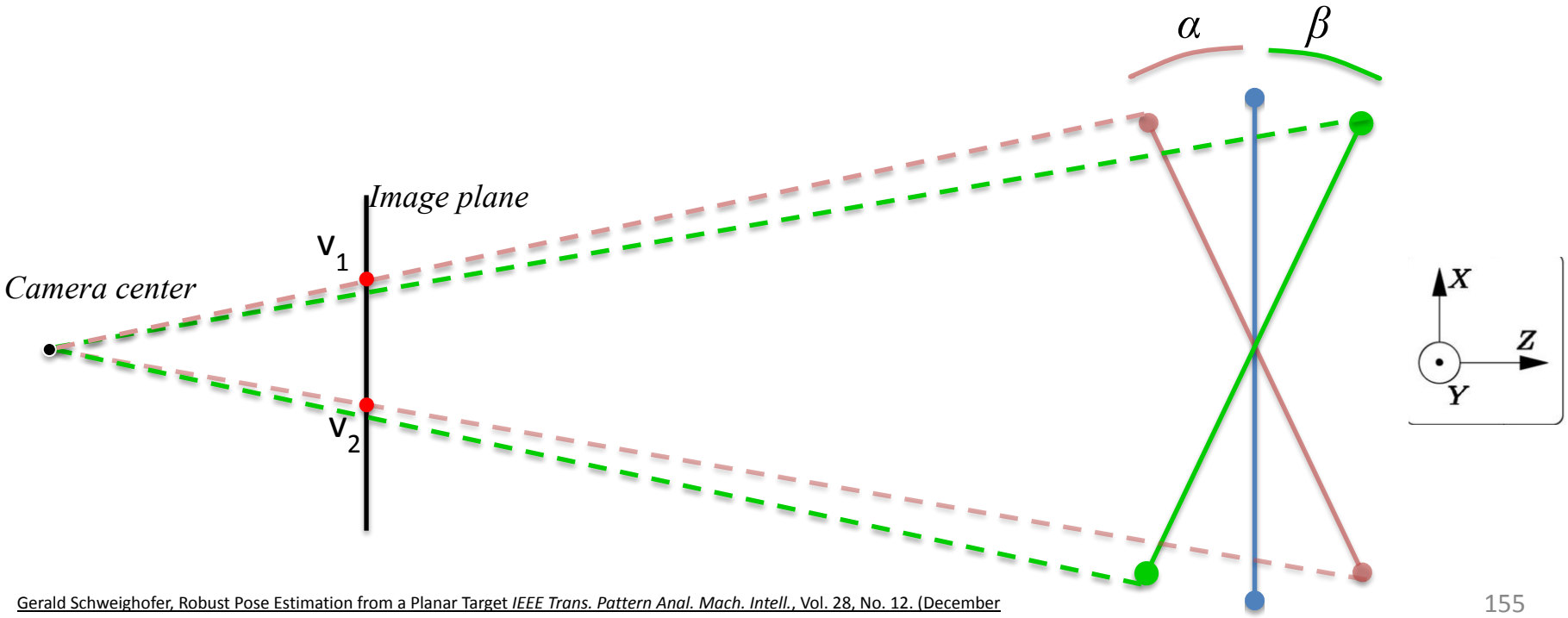


Gerald Schweighofer, Robust Pose Estimation from a Planar Target *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 28, No. 12. (December 2006), pp. 2024-2030.

Marker tracking

Pose estimation

- The Robust Planar Pose (RPP) algorithm (ArtoolkitPlus)
  - Intuitive clue: for every chosen α there may exist a second, different angle β which also leads to a local minimum of the error function



Gerald Schweighofer, Robust Pose Estimation from a Planar Target *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 28, No. 12. (December 2006), pp. 2024-2030.

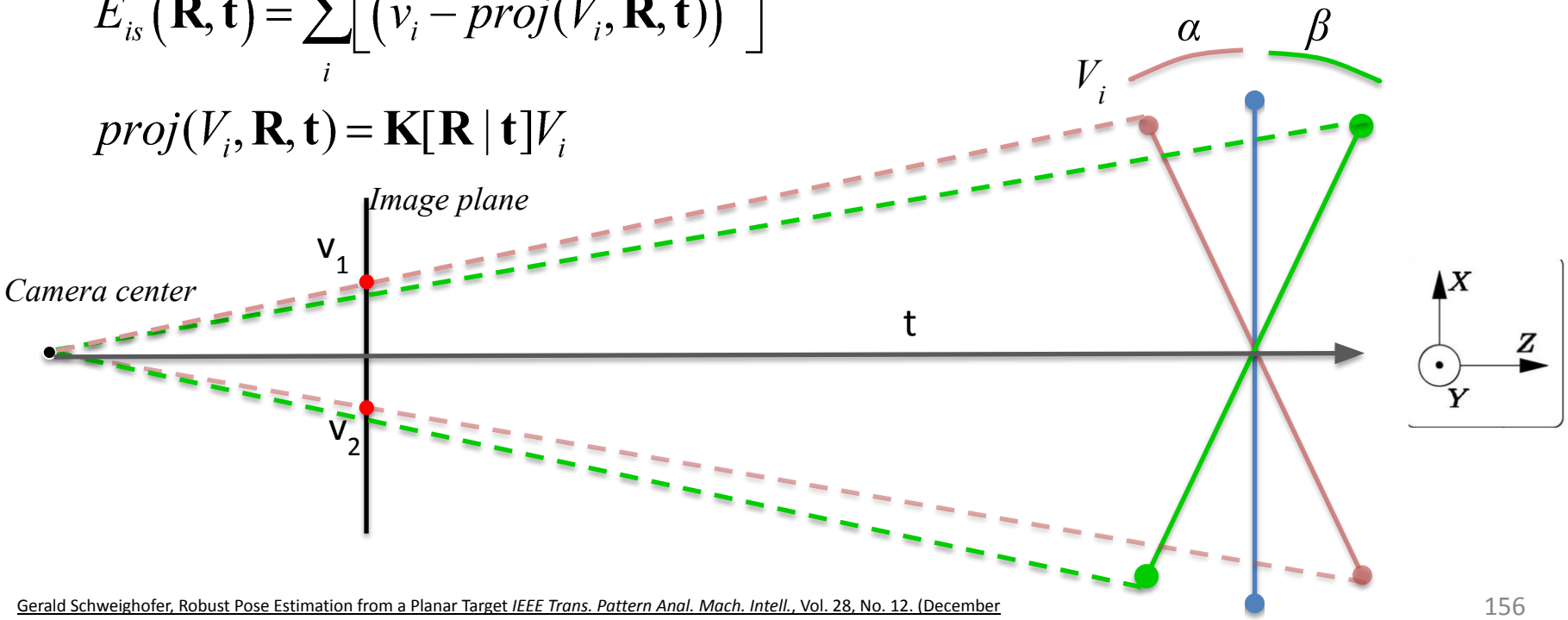
Marker tracking

Pose estimation

- The Robust Planar Pose (RPP) algorithm (ArtoolkitPlus)
  - Intuitive clue: for every chosen α there may exist a second, different angle β which also leads to a local minimum of the error function

$$E_{is}(\mathbf{R},\mathbf{t}) = \sum_i \left[\left(v_i - proj(V_i,\mathbf{R},\mathbf{t}) \right)^2 \right]$$

$$proj(V_i,\mathbf{R},\mathbf{t}) = \mathbf{K}[\mathbf{R}\,|\,\mathbf{t}]V_i$$

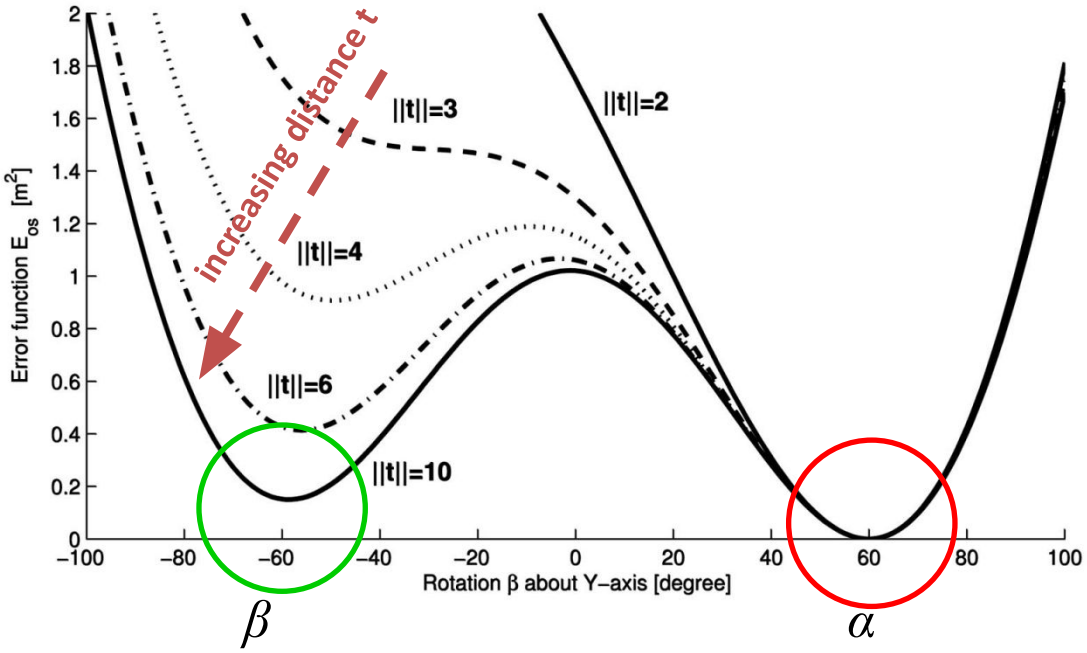


Gerald Schweighofer, Robust Pose Estimation from a Planar Target *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 28, No. 12. (December 2006), pp. 2024-2030.

Marker tracking

Pose estimation

- The Robust Planar Pose (RPP) algorithm (ArtoolkitPlus)
  - Intuitive clue: for every chosen α there may exist a second, different angle β which also leads to a local minimum of the error function



Two local minima for the reprojection error function $\mathbf{E}_{is}$ as the distance **t** of the marker varies

$$E_{is}(\mathbf{R},\mathbf{t}) = \sum_i \left[\left(v_i - proj(V_i,\mathbf{R},\mathbf{t}) \right)^2 \right]$$

$$proj(V_i,\mathbf{R},\mathbf{t}) = \mathbf{K}[\mathbf{R}\,|\,\mathbf{t}]V_i$$

Gerald Schweighofer, Robust Pose Estimation from a Planar Target *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 28, No. 12. (December 2006), pp. 2024-2030.

Marker tracking

Pose estimation

- The Robust Planar Pose (RPP) algorithm (ArtoolkitPlus)
  - Intuitive clue: for every chosen α there may exist a second, different angle β which also leads to a local minimum of the error function
- Use a first rough estimation for the pose
- Compute all the possible local minima
- Feed any iterative method with the found poses and choose the one with the smallest reprojection error

Gerald Schweighofer, Robust Pose Estimation from a Planar Target *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 28, No. 12. (December 2006), pp. 2024-2030.

Marker tracking

Pose estimation

- Non-linear least-squares optimization

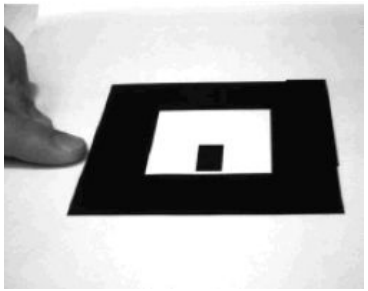
$$\min_{\mathbf{R},\mathbf{t}} \sum_i d\left(\mathbf{x}_i, \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}_i\right)$$

- Minimization of a physical, meaningful error
  - Reprojection error: distance from the measured point on the image and the reprojected 3D point
  - 6 parameters to refine (3 for orientation, 3 for position)
  - Better if more point associations are available
- Bundle adjustment
- Iterative methods may fall in local minima
  - Initial solution is important!

Marker tracking

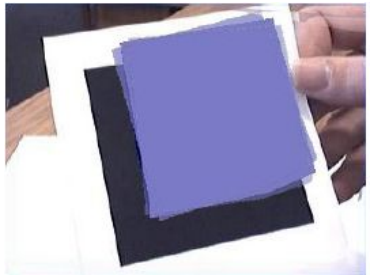
Summing up:

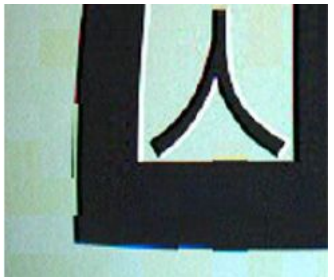
- Detect the marker
  - Edges, corners, test candidates
- Identify the marker
- Pose estimation from 4 corners
  - Homography, edge back-projection, RPP…
  - Refine the estimation with minimization
- Issues:


occlusions


blur


lighting


jittering

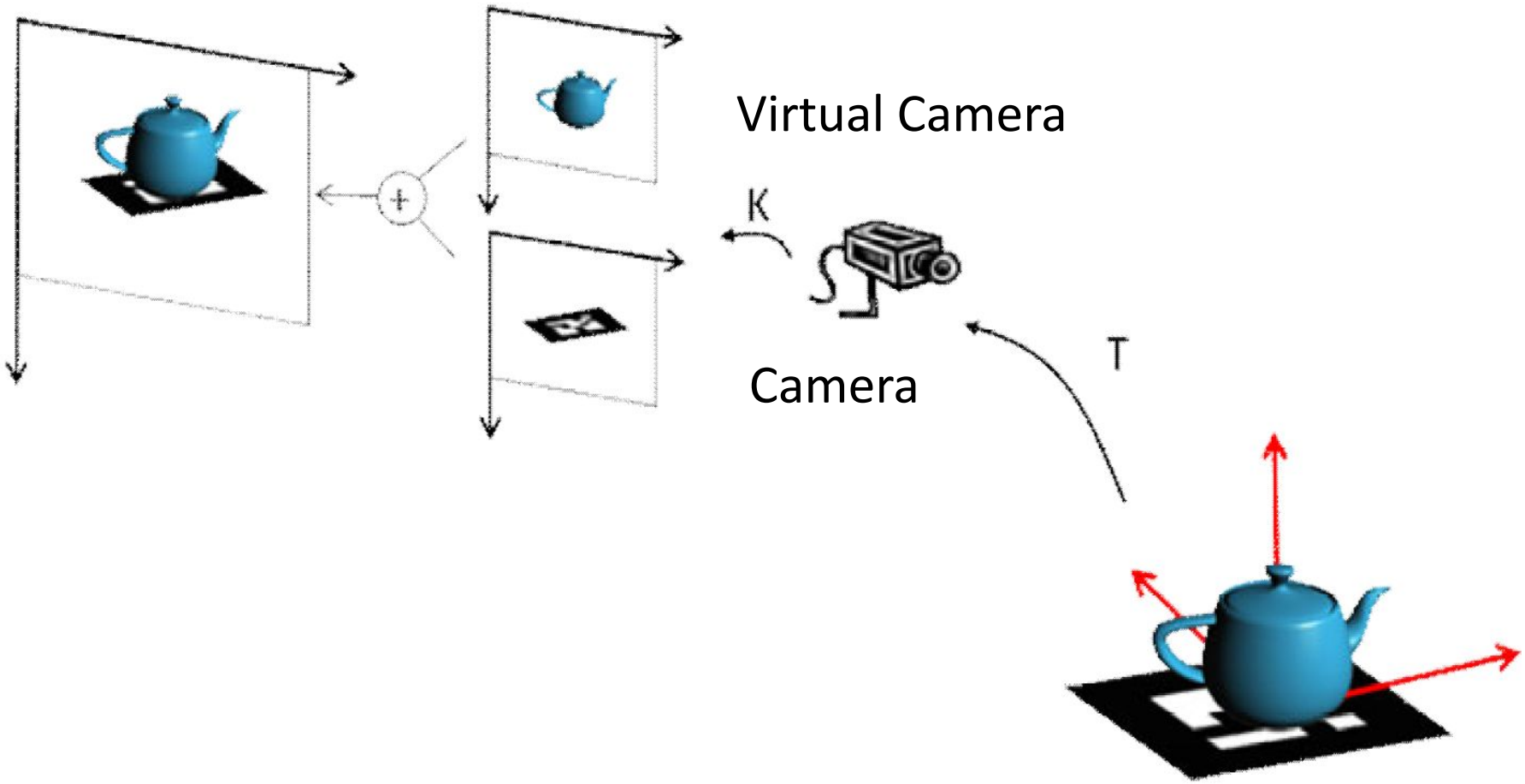

Poor quality

Marker tracking

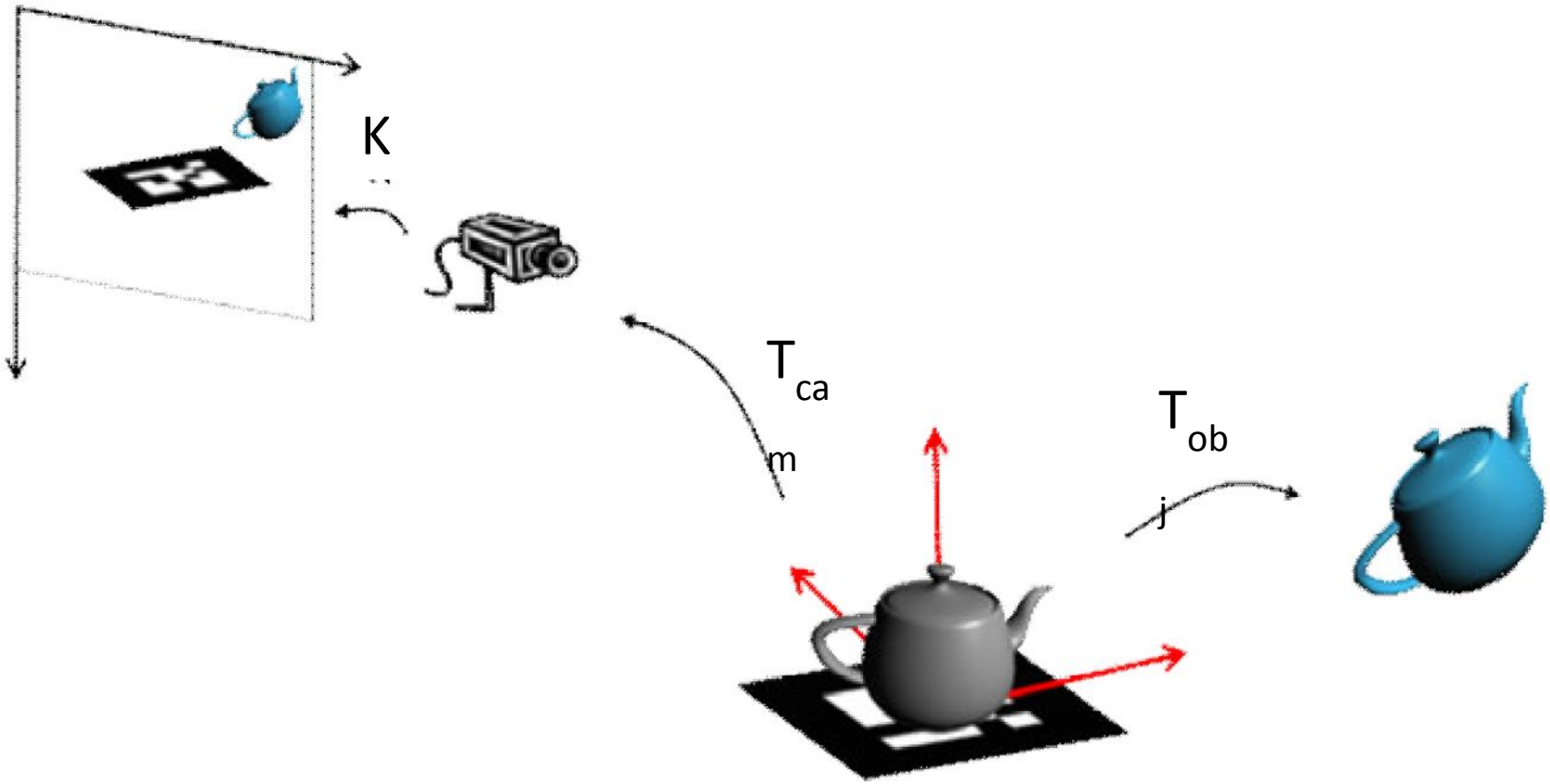
Continuous tracking:

- Use information from previous frame to “guess” the new marker position
  - Camera movement between consecutive frames is small
  - Previous pose can be used as initial guess
- It can be used to
  - Deal with occlusions
  - Deal with blur
  - Deal with markers becoming too small
- Kalman filter
  - Predict the new pose
  - Calculate the error and refine

Rendering



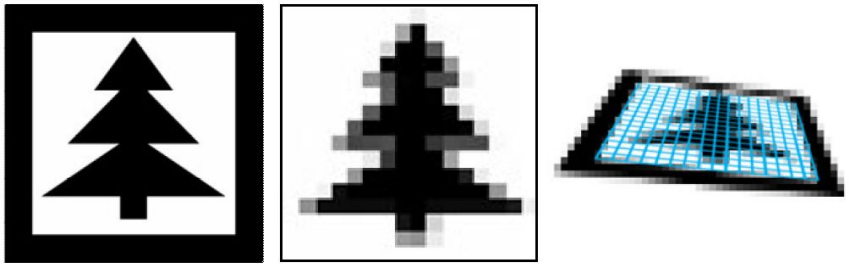
Rendering



Marker types

Template markers

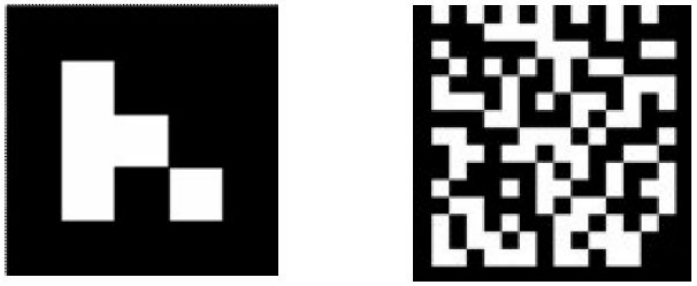
- Black and white with an image inside



- The image can be identified by template matching against a database of markers
- It does not scale up with the number of markers

Data matrix marker

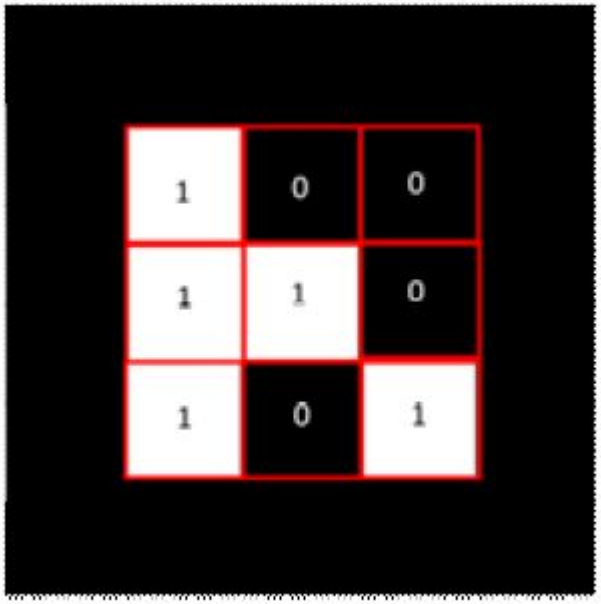
- Black and white data cells that encode the information (marker id, text, hyperlink…)



- Easier to detect and identify
- Error detection and correction
- Trade-off needed between detection robustness and encoded data

Data matrix markers

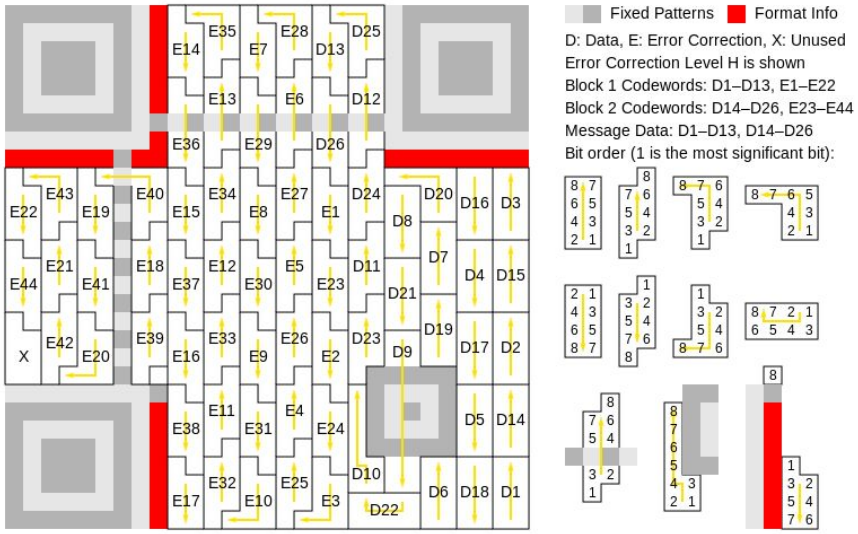
- The data is directly encoded in the marker



marker ID: 100 110 101 (= 309)

- Besides encoding information, part of the bits for error detection and correction. (ArTag)

QR Barcodes

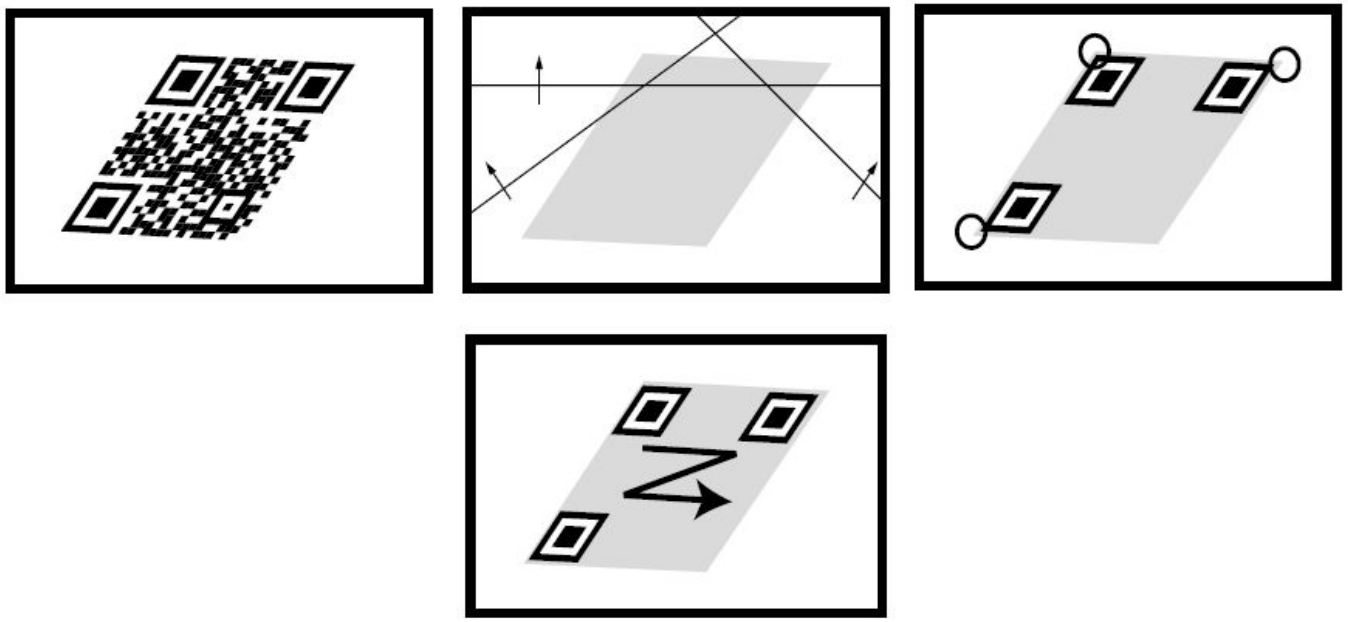


http://www.datagenetics.com/blog/november12013/index.html

- Good identification
  - Error correction
  - configurable error correction levels: higher error correction level, less storage capacity, and viceversa

QR Barcodes

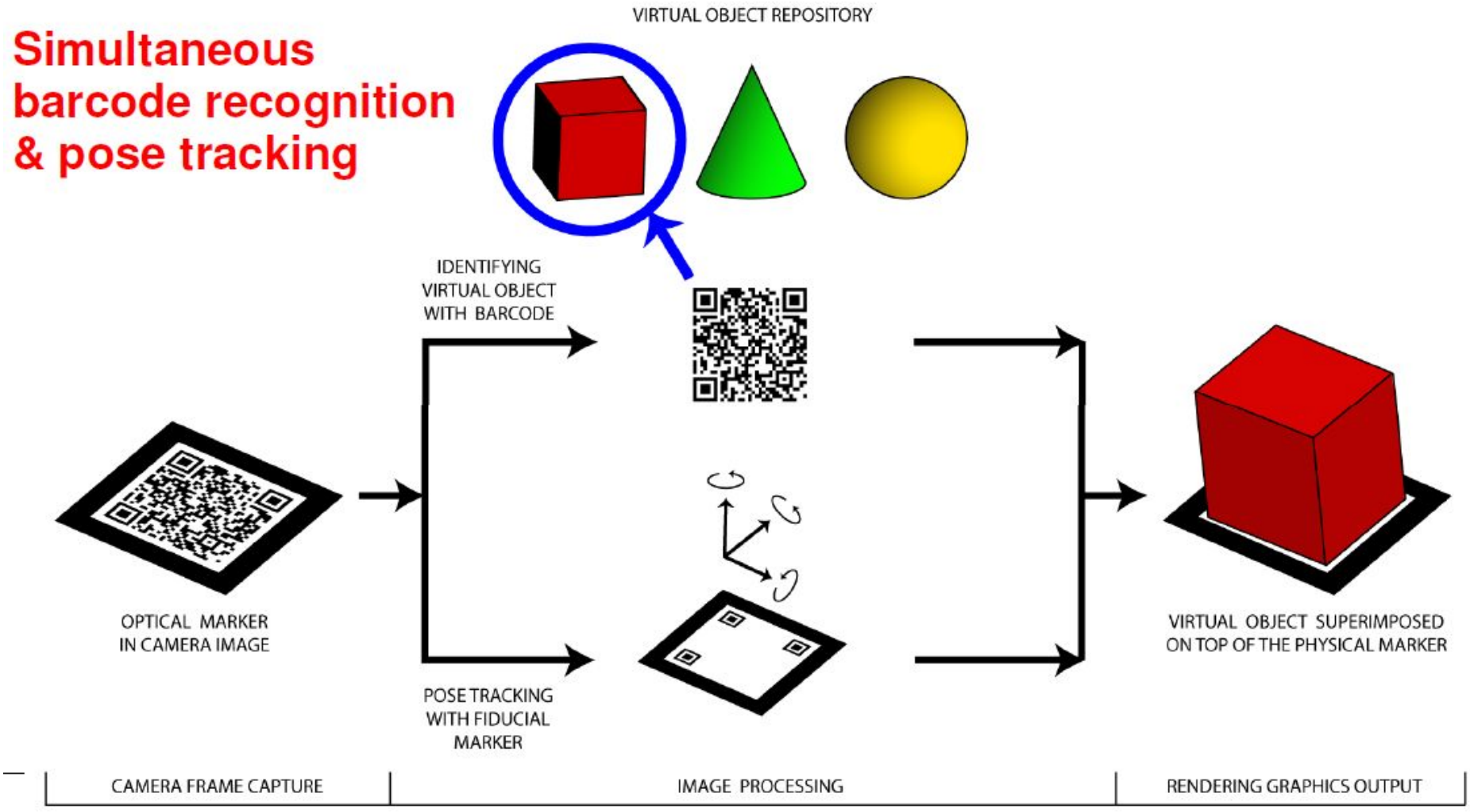
- Not meant for pose estimation
- Use 3 points to unwarp the image
  - Affine transformation does not compensate for severe perspective distortion
- They require a large area of the image for correct decoding



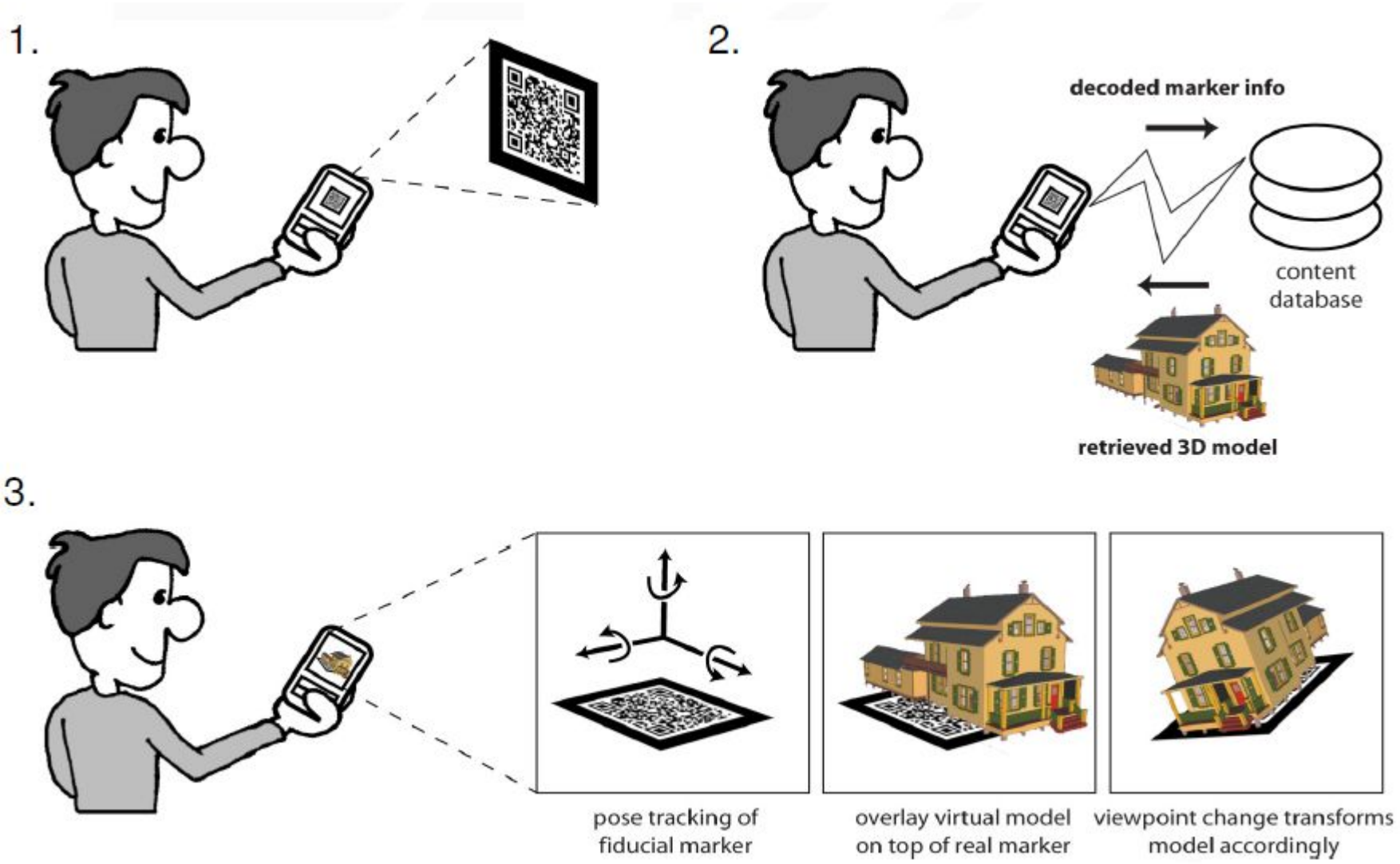
QR Detection – Examples

http://www.datagenetics.com/blog/november12013/index.html

Data Matrix + QR Code



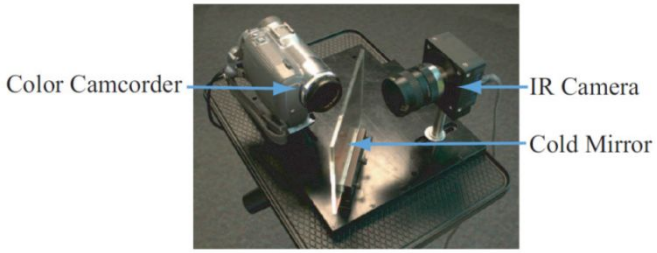
Data Matrix + QR Code



Other markers

- Image markers
  - image as marker
  - Features (sift, surf…) detection and matching
  - estimation and decomposition of H
  - Robust to occlusion (sparse detection)

- Infrared markers



Final remarks

Pros

- Easy setup and low cost
  - Even for mobile devices
- Works well with
  - Large movement of the camera
  - Dynamic environments (as long as the marker is visible)
  - Texture-less scenes
- Encodes additional info
- Always correct scale for 3D object

Cons

- Requires setup
- Markers must be always (at least partially) visible
- May require marker digital removal

References

- D.W.F. Van Krevelen and R.Poelman, "A survey of Augmented Reality Technologies, Applications and Limitations", The international journal of Virtual Reality, 9(2):1-20
- V. Lepetit and P. Fua. "Monocular model-based 3D tracking of rigid objects"

PnP

- M. Fischler and R. Bolles "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography"
- A. Zisserman and R. Hartley "Multiple View Geometry"

Model-based tracking

- T. Drummond, R. Cipolla "Real-Time Visual Tracking of Complex Structures"

Marker tracking

- Sanni Siltanen, "Theory and applications of marker-based augmented reality"
- M. Fiala "ARTag, An Improved Marker System Based on ARToolkit"