

# Augmented Reality

## Lecture 10 – camera calibration and HMD calibration

Manuela Chessa – [manuela.chessa@unige.it](mailto:manuela.chessa@unige.it)

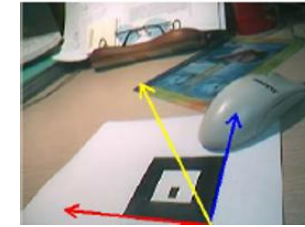
Fabio Solari – [fabio.solari@unige.it](mailto:fabio.solari@unige.it)

# Summary

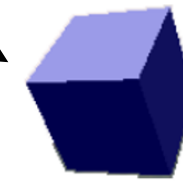
- Camera calibration (*single view geometry*)
  - Camera model
  - Intrinsic and extrinsic parameters
  - Camera projection matrix
  - Camera calibration
- Display calibration (*optical see-through head-mounted display*)
  - Single Point Active Alignment Method (SPAAM)
  - HMD Calibration Using a Pointing Device
- Issues and error sources in AR registration

# Calibration and Registration

- Using a **tracking system** creates the need to handle **multiple coordinate systems**.
- These coordinate systems need to be *reconciled* with each other, so that a **correct overlay** of **virtual** objects on tracked **physical objects** is ensured.
- This process, which is called **registration**, is required to convert the poses from tracking into the coordinate system of the rendering application.
- Registration is also necessary to align the rendering camera with respect to a tracked display



World coordinate system



Virtual coordinate system



Correct overlay needs registration of coordinate systems

# Calibration and Registration

- **Registration** in AR means that several components **need calibration**.
- We begin by examining the calibration of the **camera's** internal parameters and lens distortion.
- Then turn our attention to **displays**, discussing the calibration of *optical see-through head-mounted displays* without and with the help of an additional pointing device.

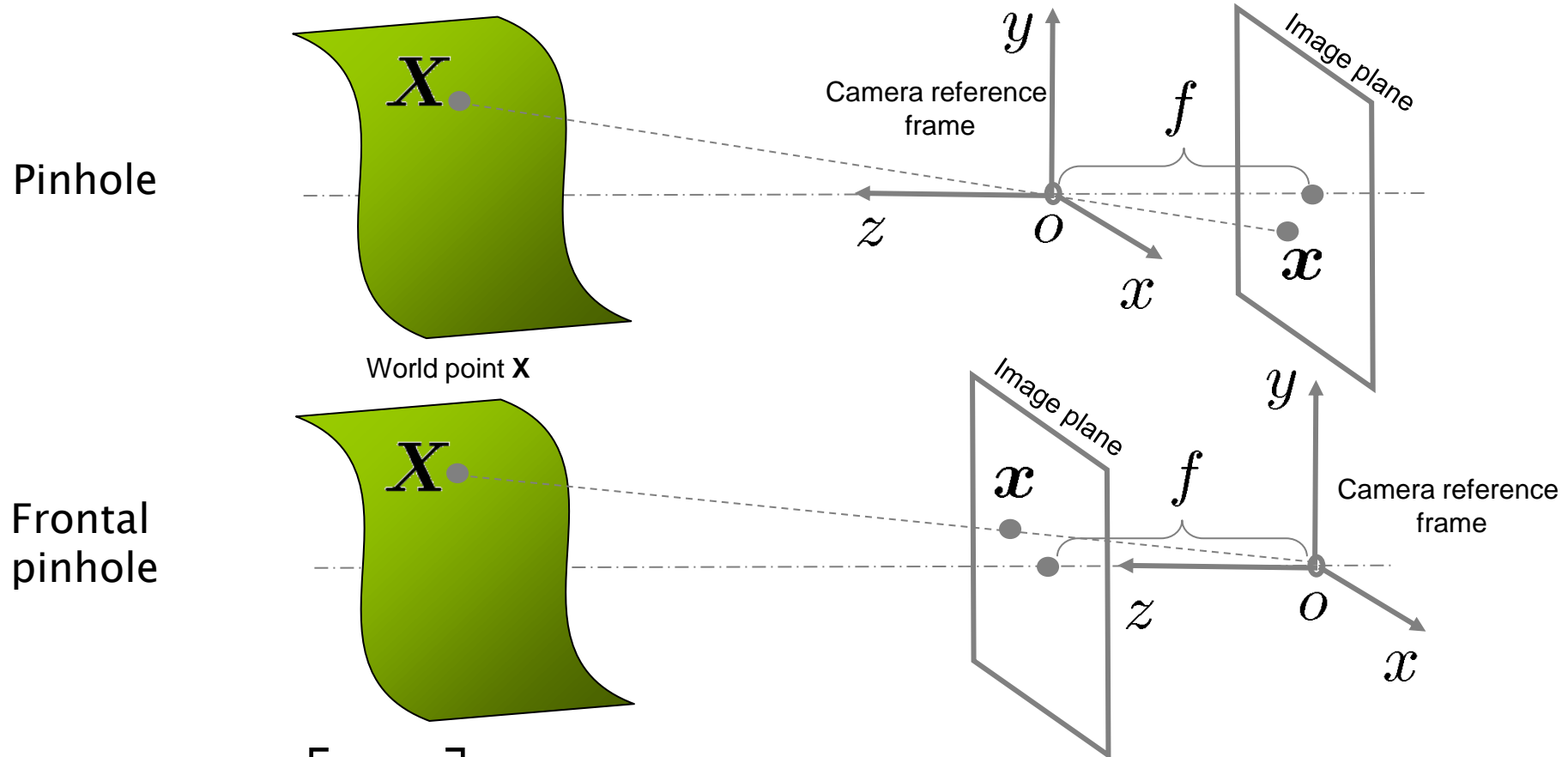


**Università  
di Genova**

**DIBRIS** DIPARTIMENTO  
DI INFORMATICA, BIOINGEGNERIA,  
ROBOTICA E INGEGNERIA DEI SISTEMI

# Camera calibration

# Pinhole camera model



$$X = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \rightarrow x = \begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} X \\ Y \end{bmatrix}$$

# Pinhole camera model

2-D coordinates  $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} X \\ Y \end{bmatrix}$

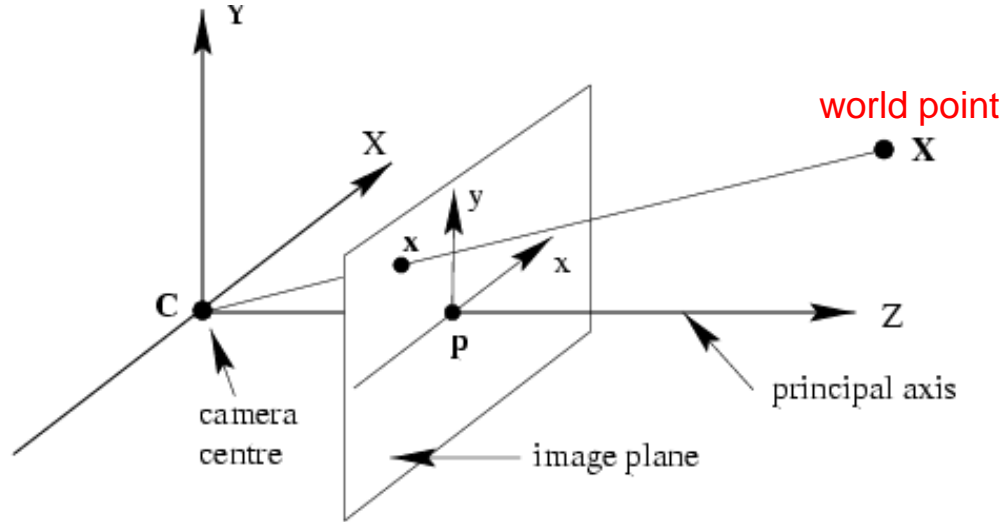
Homogeneous coordinates

$$\mathbf{x} \rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix}, \quad \mathbf{X} \rightarrow \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix},$$

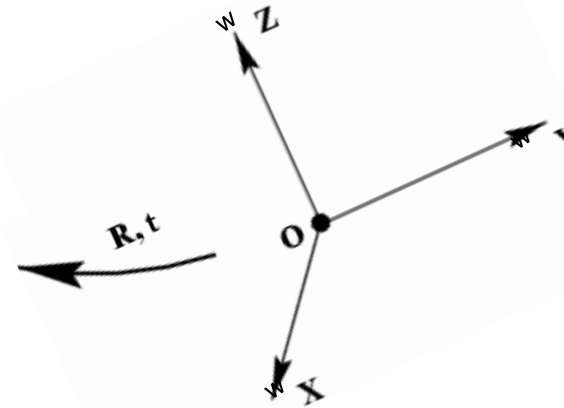
$$\begin{array}{c} \nearrow Z \\ w, \lambda, s \end{array} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{K_f} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\Pi_0} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

# Pinhole camera model

camera coordinate system



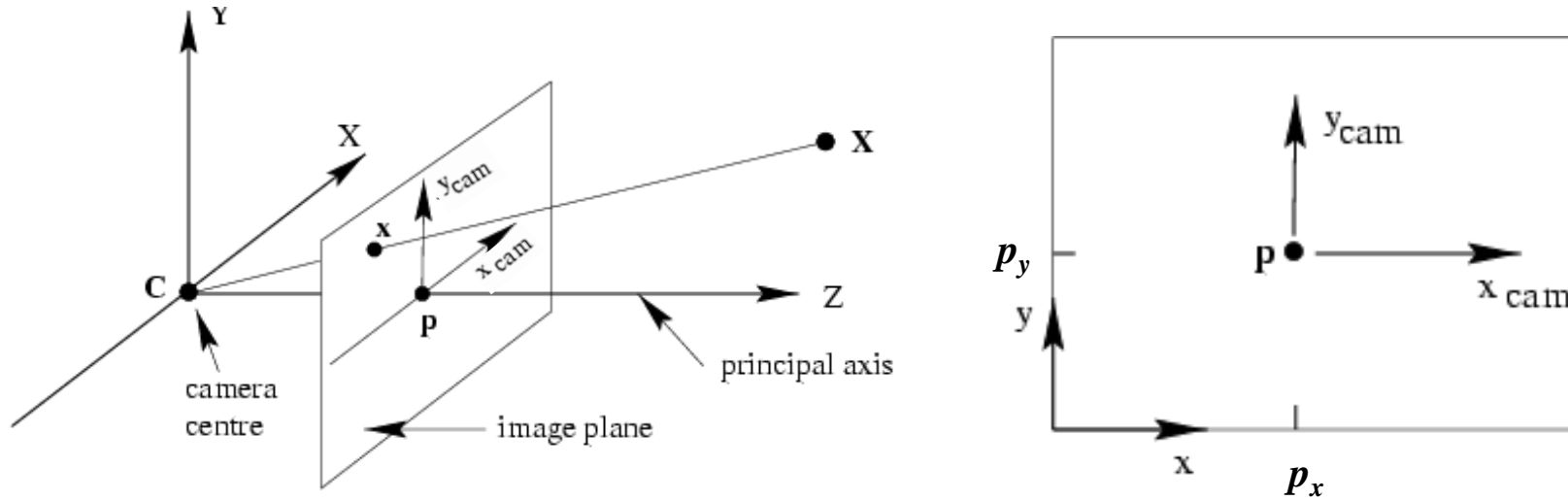
world coordinate system



- **Camera calibration**: figuring out
  1. transformation from metric coordinates (camera) to pixels coordinates (image)
  2. transformation from *world* coordinate system to *camera* coordinate system

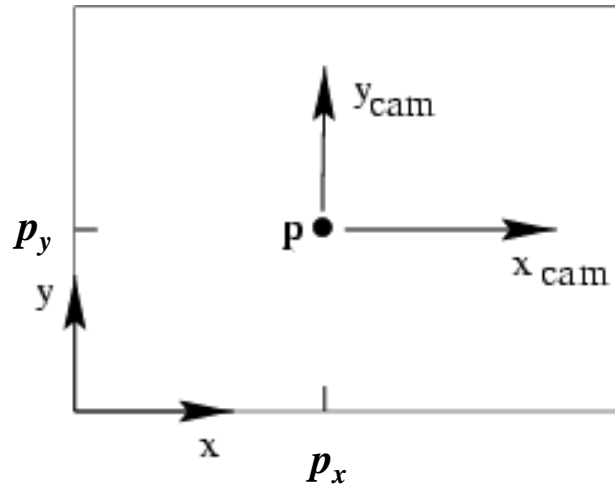


# Image coordinate system



- **Principal point ( $p$ ):** point where principal (optical) axis, the  $z$ -axis, intersects the image plane.
- **Normalized coordinate system:** origin of the image is at the principal point;  $x$  and  $y$  axes of the image plane are parallel to  $X$  and  $Y$  axes of the camera reference system.
- **Image coordinate system:** origin is in the corner.  
(not pixels)

# Principal point offset

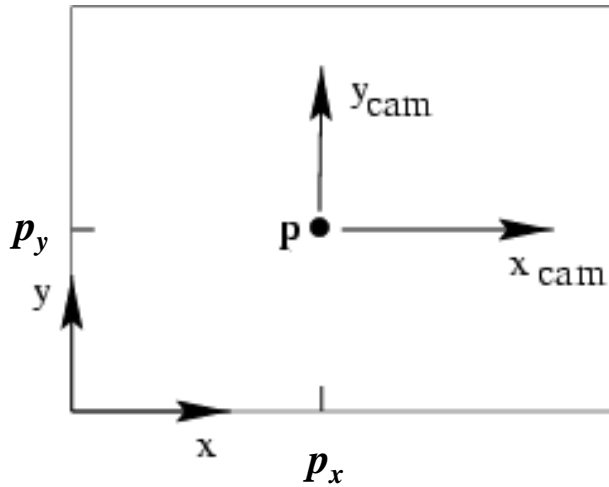


We want the principal point to map to  $(p_x, p_y)$  instead of  $(0,0)$

$$(X, Y, Z) \rightarrow (f X / Z + p_x, f Y / Z + p_y) = (x + p_x, y + p_y)$$

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \xrightarrow{\frac{1}{Z}} \begin{pmatrix} f X + Z p_x \\ f Y + Z p_y \\ Z \end{pmatrix} = \begin{bmatrix} f & p_x & 0 \\ & f & p_y \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

# Principal point offset



principal point:  $(p_x, p_y)$

$$\mathbf{x} = \mathbf{P}_0 \mathbf{X} \quad \mathbf{P}_0 = \mathbf{K} [\mathbf{I}, 0]$$

$$\frac{1}{Z} \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ & 1 & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

s: the skew factor s is needed only if the image directions x and y are not perpendicular

$$\mathbf{K} = \begin{bmatrix} f & s & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}$$

**calibration matrix K or  $\mathbf{K}_f$**

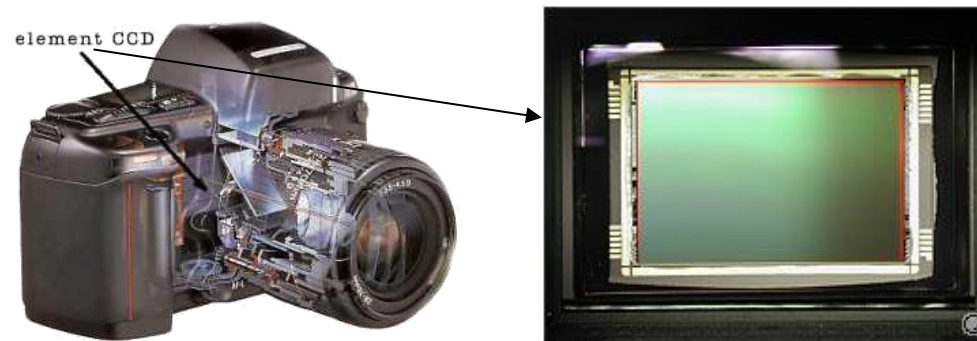
# Pixel coordinates

$$K = \begin{matrix} & K_s & \\ \begin{bmatrix} m_x & & \\ & m_y & \\ & & 1 \end{bmatrix} & \begin{matrix} \text{pixels/m} \end{matrix} & \begin{matrix} K_f \\ \begin{bmatrix} f & s & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} \end{matrix} \begin{matrix} \text{meters} \end{matrix} = \begin{matrix} \begin{bmatrix} f_x & s_x & u_o \\ & f_y & v_o \\ & & 1 \end{bmatrix} \\ \text{pixels} \end{matrix}$$

The parameters  $f_x$  and  $f_y$  describe the focal length of the camera, scaled by the size of the pixel in the directions  $x$  and  $y$ , respectively

from metric coordinates (camera) to pixels coordinates (image)

$m_x$  pixels per meter in horizontal direction,  
 $m_y$  pixels per meter in vertical direction



# Calibration Matrix and Camera Model

Pinhole camera

$$\lambda \mathbf{x} = K_f \Pi_0 \mathbf{X}$$

Pixel coordinates

$$\mathbf{x}' = K_s \mathbf{x}$$

$$\lambda \mathbf{x}' = K_s K_f \Pi_0 \mathbf{X} = \underbrace{\begin{bmatrix} f s_x & f s_\theta & o_x \\ 0 & f s_y & o_y \\ 0 & 0 & 1 \end{bmatrix}}_{K} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\Pi_0} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Calibration matrix  
(intrinsic parameters)

$$K = K_s K_f \quad \Pi_0$$

Projection matrix

$$\Pi = [K, 0] \in \mathbb{R}^{3 \times 4}$$

Camera model

$$\lambda \mathbf{x}' = K \Pi_0 \mathbf{X} = \Pi \mathbf{X}$$

or

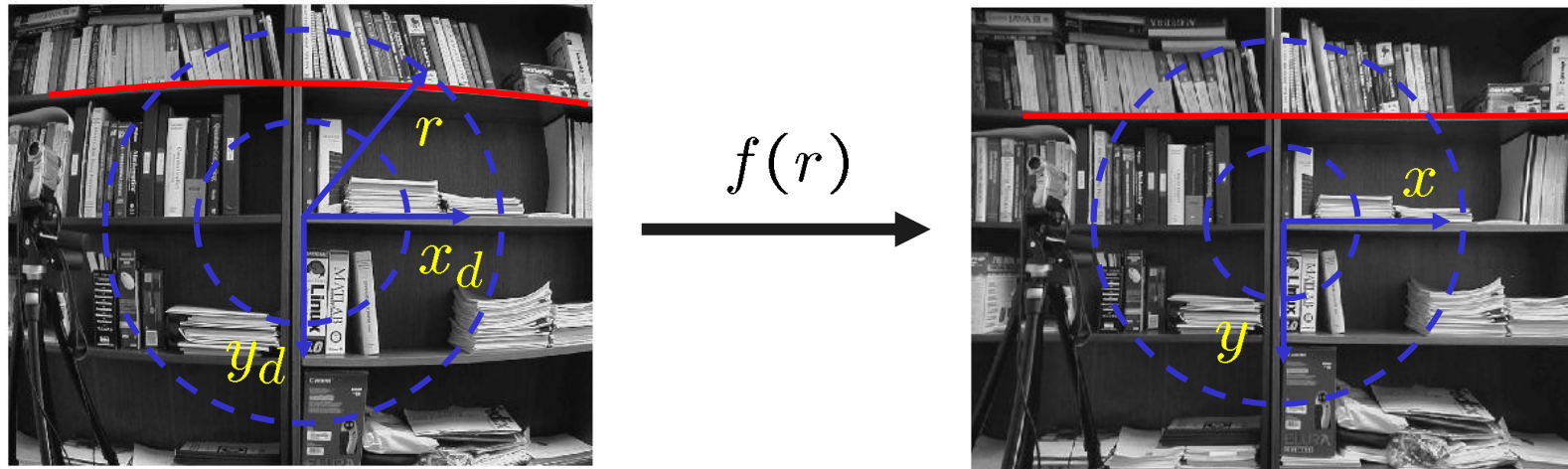
$$\lambda \mathbf{x} = \mathbf{P}_0 \mathbf{X}_{\text{cam}}$$

# Camera parameters: Lens Distortion Model

**Non-linear effects** (*for short focal length and cheap cameras*):

- Radial distortion
- Tangential distortion

Example: Nonlinear transformation along the radial direction



**Distortion correction:**  
make lines straight  
(image warping)

# Camera parameters: Radial Distortion (lenses)

Compute the corrected image point:

$$(1) \quad \begin{aligned} x' &= x/z \\ y' &= y/z \end{aligned}$$

$$(2) \quad \begin{aligned} x'' &= x'(1 + k_1 r^2 + k_2 r^4) + 2p_1 x' y' + p_2 (r^2 + 2x'^2) \\ y'' &= y'(1 + k_1 r^2 + k_2 r^4) + p_1 (r^2 + 2y'^2) + 2p_2 x' y' \end{aligned}$$

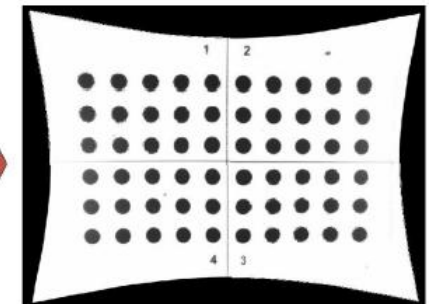
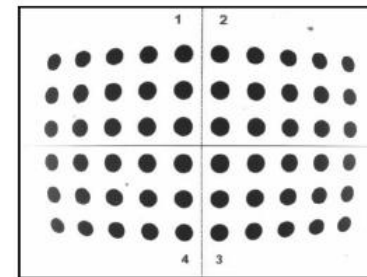
where  $r^2 = x'^2 + y'^2$   $k_1, k_2$  : radial distortion coefficients

$p_1, p_2$  : tangential distortion coefficients

$$(3) \quad \begin{aligned} u &= f_x \cdot x'' + c_x \\ v &= f_y \cdot y'' + c_y \end{aligned}$$

Radial distortion: for conventional cameras, considering a single coefficient  $k_1$  is usually satisfactory. In contrast, for wide-angle lenses, it may be necessary to consider two coefficients  $k_1$  and  $k_2$ .

**Distortion correction:**  
make lines straight  
(image warping)

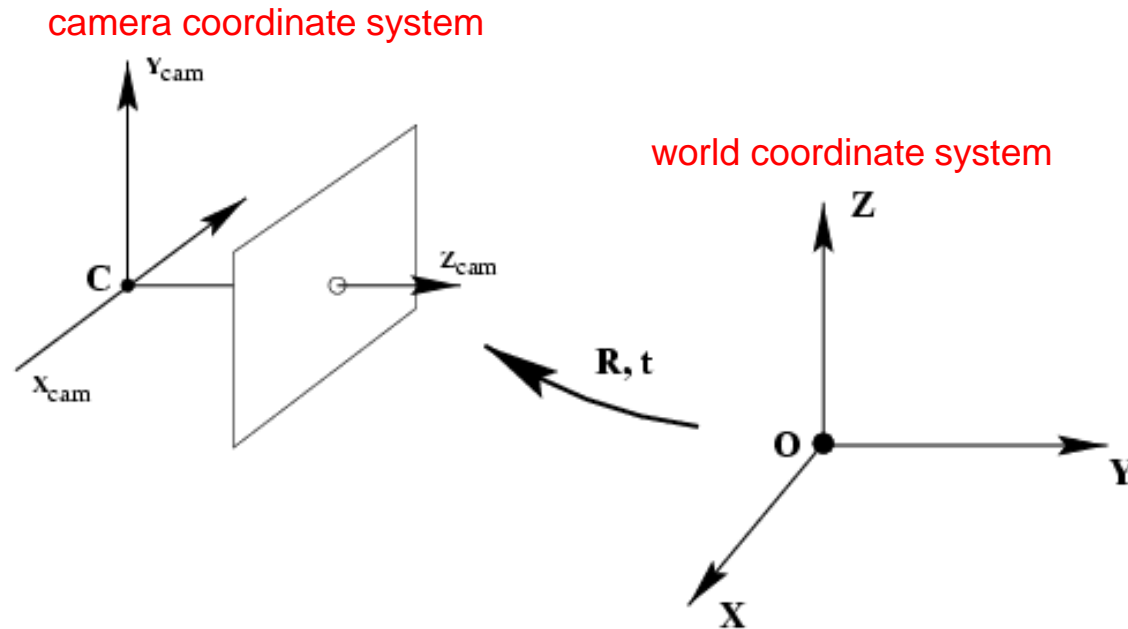


Calibration  
(off-line)



correction  
(on-line)

# Camera rotation and translation



- In general, the **camera coordinate frame** will be related to the **world coordinate frame** by a **rotation R** and a **translation t**

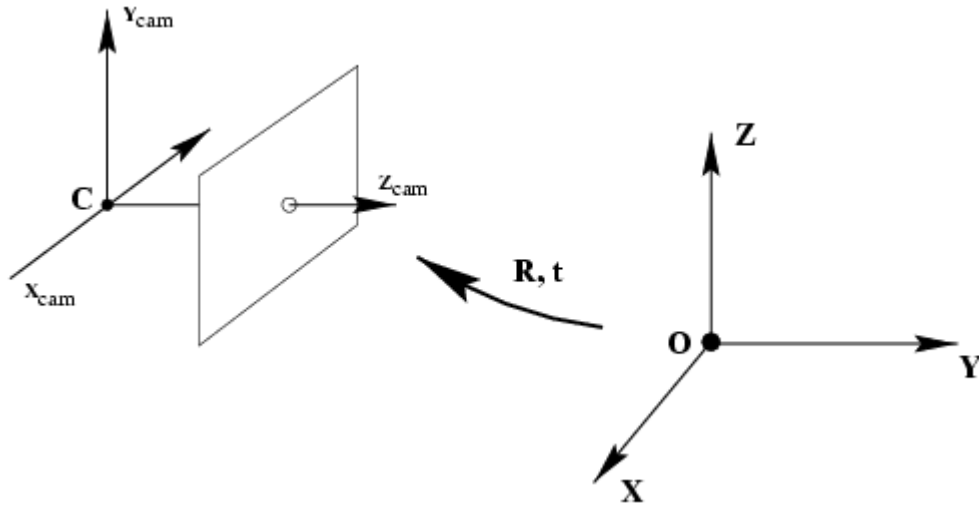
- Transformation **from world to camera** coordinate system (rigid-body motion):

$$X_{\text{cam}} = RX + t$$

*(non-homogeneous coordinates)*



# Camera rotation and translation



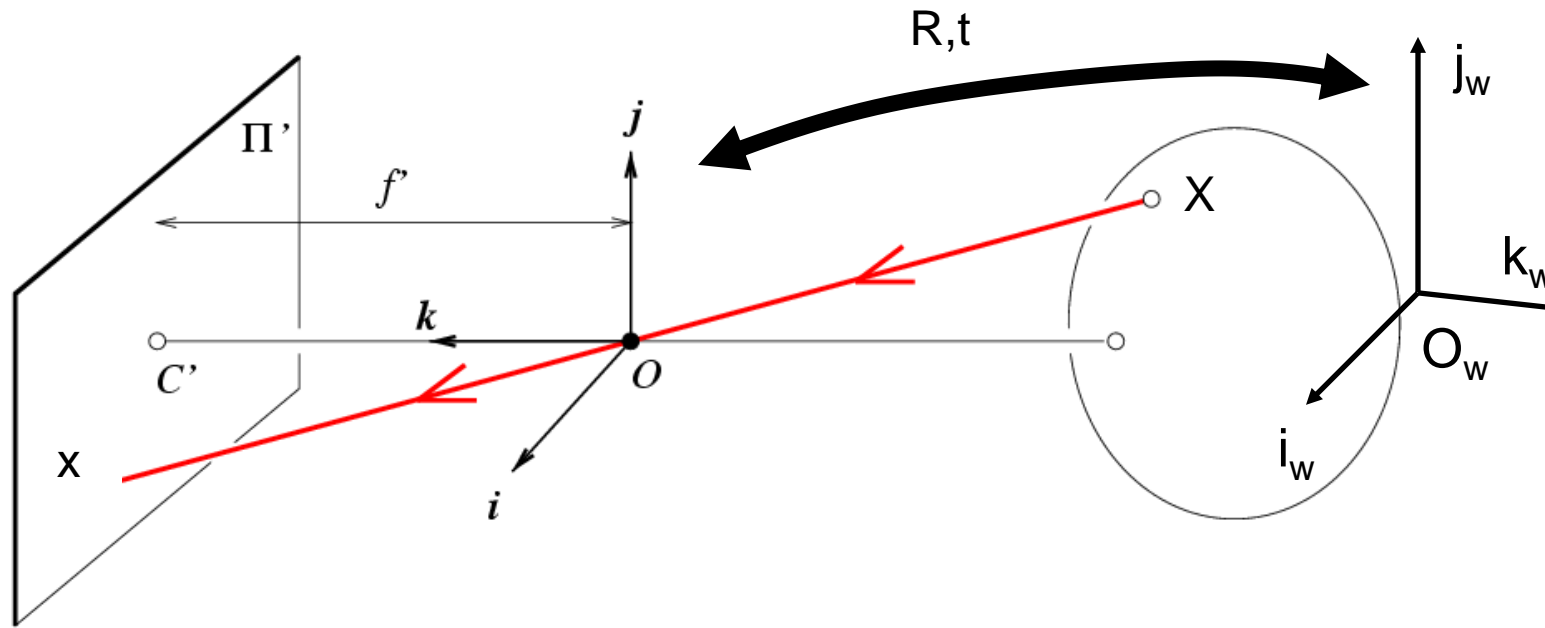
$$X_{\text{cam}} = RX + t$$

$$X_{\text{cam}} = \begin{pmatrix} X_{\text{cam}} \\ 1 \end{pmatrix} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ 1 \end{pmatrix} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} X$$

$$\lambda X = P_0 X_{\text{cam}} = K[I \mid 0] X_{\text{cam}} = K[R \mid t] X = MX$$

$$M = K[R \mid t],$$

# Camera (projection) matrix



$$\lambda \mathbf{x} = \overbrace{\mathbf{K}}^{\text{Intrinsic Matrix}} \underbrace{\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}}_{\text{Extrinsic Matrix}} \mathbf{X}$$

$\mathbf{x}$ : Image Coordinates:  $(u, v, 1)$

$\mathbf{K}$ : Intrinsic Matrix (3x3)

$\mathbf{R}$ : Rotation (3x3)

$\mathbf{t}$ : Translation (3x1)

} Extrinsic Matrix

$\mathbf{X}$ : World Coordinates:  $(X, Y, Z, 1)$

# How to calibrate the camera? (also called “camera resectioning”)

$$s\mathbf{X} = \mathbf{K}[\mathbf{R} \quad \mathbf{t}] \mathbf{X}$$

$$s\mathbf{X} = \mathbf{M}\mathbf{X}$$

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Pixel coordinates

World coordinates

Unknowns

Unknowns

Translation vector

Rotation matrix

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} \alpha & \gamma & u_0 & 0 \\ 0 & \beta & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Intrinsic parameters

Pixel coordinates

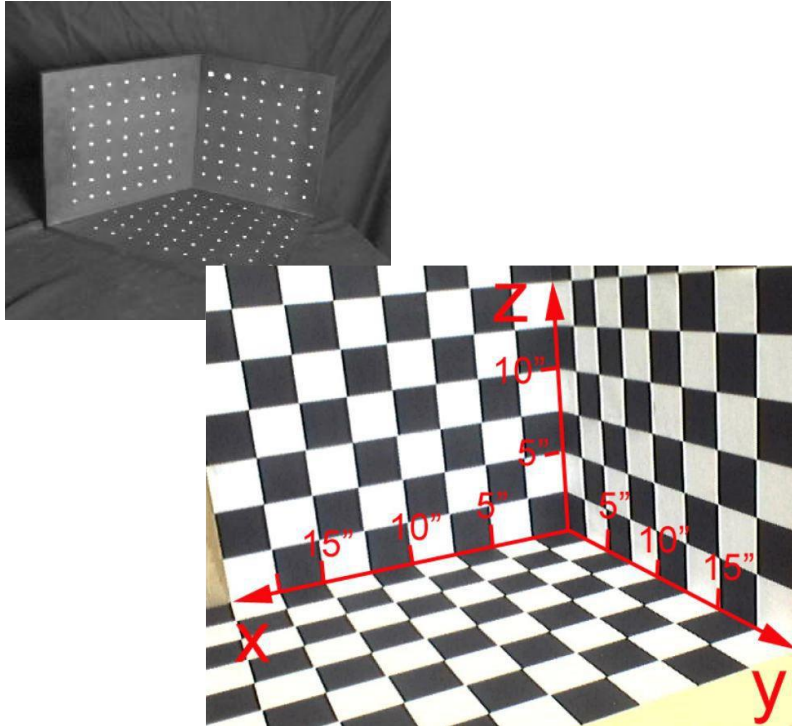
World coordinates

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} \alpha & \gamma & u_0 & 0 \\ 0 & \beta & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

3D location of the point in Camera coordinates

# Calibrating the Camera

Calibration rig  
(i.e. a scene with **known** geometry)



Known 2D  
image coords



Known 3D  
world locations



$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{matrix} \mathbf{M} \\ \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \end{matrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$



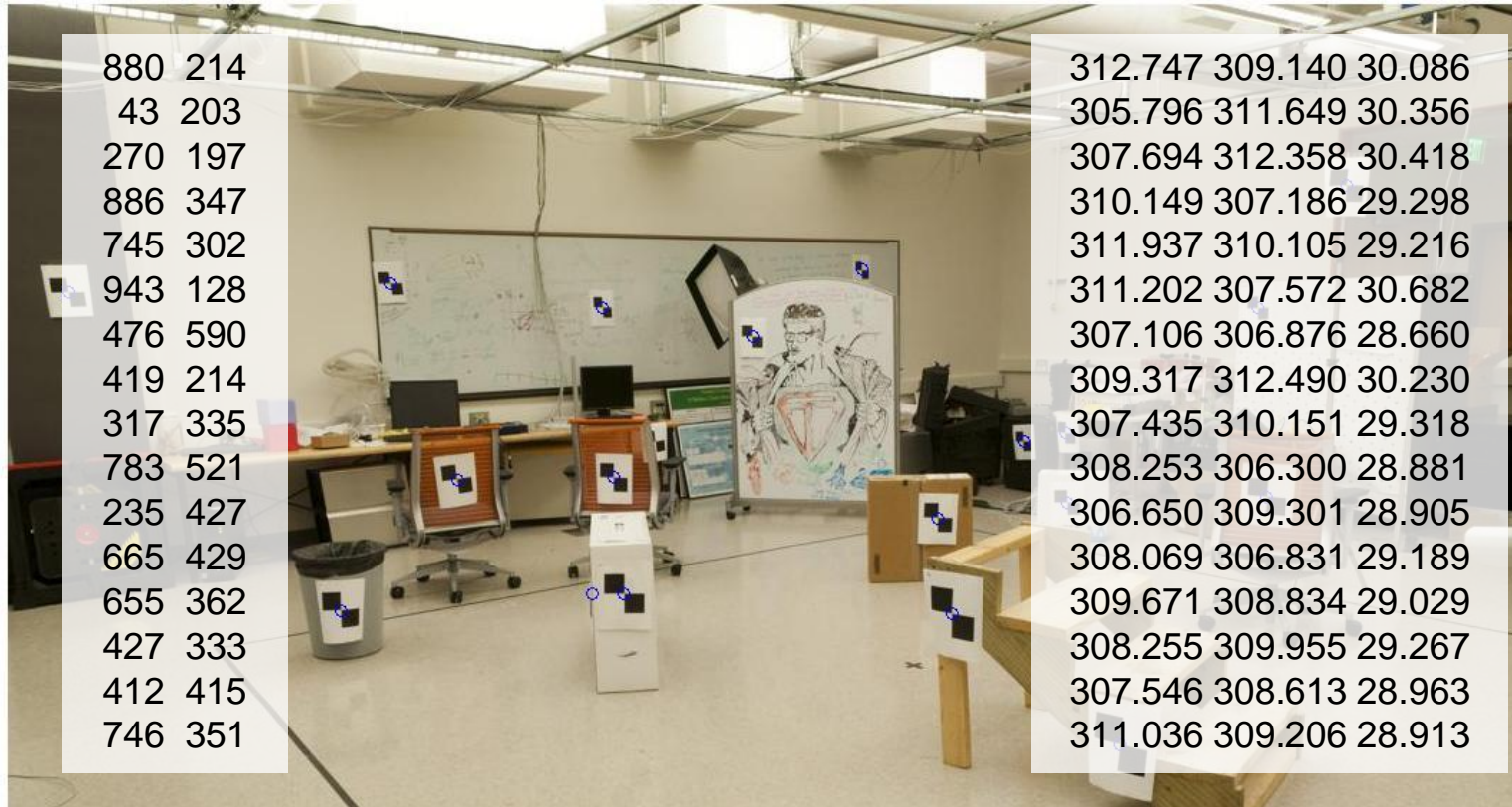
Unknown Camera Parameters

Given  $n$  points with known 3D coordinates  $X_i$  and known image projections  $x_i$ , to estimate the camera parameters (**least squares solution**)

# Calibrating the Camera

Known 2D  
image coords

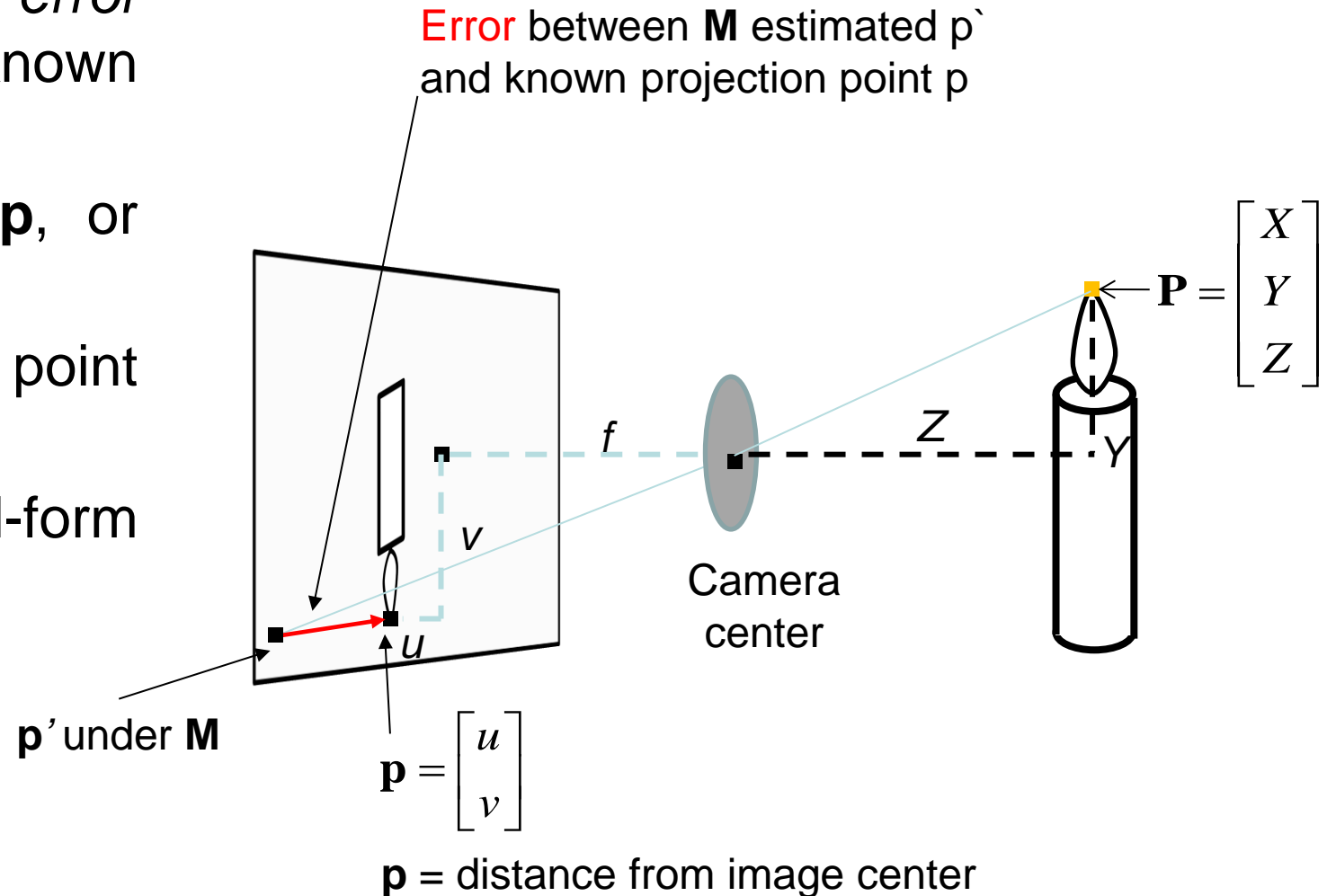
Known 3D  
world locations



Note: for coplanar points, we will get degenerate solutions.  
*However, **homography** (see next lecture)!*

# What is least squares doing?

- Given 3D point evidence, find best **M** which *minimizes the error* between estimate ( $\mathbf{p}'$ ) and known corresponding 2D points ( $\mathbf{p}$ ).
- Best **M** occurs when  $\mathbf{p}' = \mathbf{p}$ , or when  $\mathbf{p}' - \mathbf{p} = 0$
- Form these equations from all point evidence
- Solve for model via closed-form regression



# Calibrating the Camera: Direct linear method

Known 2d  
image coords

Unknown Camera Parameters

$$\underbrace{\begin{bmatrix} su \\ sv \\ s \end{bmatrix}}_{\mathbf{p}} = \underbrace{\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix}}_{\mathbf{p}'} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix},$$

Known 3d  
locations

*Writing as a system of  
linear equations  $\mathbf{A}\mathbf{m}=\mathbf{0}$*

First, work out  
where X,Y,Z  
projects to under  
candidate **M**

$$su = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$

$$sv = m_{21}X + m_{22}Y + m_{23}Z + m_{24}$$


$$s = m_{31}X + m_{32}Y + m_{33}Z + m_{34}$$

**Two equations  
per 3D point  
correspondence**

$$u = \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}$$

$$v = \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}$$

# Calibrating the Camera: Direct linear method

 Unknown Camera Parameters

$$\begin{array}{c} \text{Known 2d} \\ \text{image coords} \end{array} \begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{array}{c} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \\ \text{Known 3d} \\ \text{locations} \end{array}$$

Next, rearrange into form where all **M** coefficients are individually stated in terms of X,Y,Z,u,v.

-> Allows us to form *lsq matrix*.

$$u = \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}$$

$$v = \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}$$

$$(m_{31}X + m_{32}Y + m_{33}Z + m_{34})u = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$


$$(m_{31}X + m_{32}Y + m_{33}Z + m_{34})v = m_{21}X + m_{22}Y + m_{23}Z + m_{24}$$

$$m_{31}uX + m_{32}uY + m_{33}uZ + m_{34}u = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$

$$m_{31}vX + m_{32}vY + m_{33}vZ + m_{34}v = m_{21}X + m_{22}Y + m_{23}Z + m_{24}$$



# Calibrating the Camera: Direct linear method


Unknown Camera Parameters

$$\begin{array}{c} \text{Known 2d} \\ \text{image coords} \end{array}
 \begin{bmatrix} su \\ sv \\ s \end{bmatrix}
 =
 \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix}
 \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}
 \begin{array}{c} \text{Known 3d} \\ \text{locations} \end{array}$$

Next, rearrange into form  
where all **M** coefficients are  
individually stated in terms  
of X,Y,Z,u,v.

-> Allows us to form lsq  
matrix.

$$m_{31}uX + m_{32}uY + m_{33}uZ + m_{34}u = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$

$$m_{31}vX + m_{32}vY + m_{33}vZ + m_{34}v = m_{21}X + m_{22}Y + m_{23}Z + m_{24}$$

$$0 = m_{11}X + m_{12}Y + m_{13}Z + m_{14} - m_{31}uX - m_{32}uY - m_{33}uZ - m_{34}u$$

$$0 = m_{21}X + m_{22}Y + m_{23}Z + m_{24} - m_{31}vX - m_{32}vY - m_{33}vZ - m_{34}v$$

# Calibrating the Camera: Direct linear method

Unknown Camera Parameters

Known 2d image coords

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Known 3d locations

- Since **every point** gives **two equations**, we need at least **6 non-coplanar points** to solve
- Solve for m's entries using total linear least-squares
- Method –  **$Ax=0$**  form (*eigenvector with smallest eigenvalue*)

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1 X_1 & -u_1 Y_1 & -u_1 Z_1 & -u_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1 X_1 & -v_1 Y_1 & -v_1 Z_1 & -v_1 \\ & & & & & & \vdots & & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_n X_n & -u_n Y_n & -u_n Z_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_n X_n & -v_n Y_n & -v_n Z_n & -v_n \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \cdot \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{aligned} [U, S, V] &= \text{svd}(A); \\ M &= V(:, \text{end}); \\ M &= \text{reshape}(M, [], 3)'; \end{aligned}$$

# Calibrating the Camera

- *Homogeneous least squares*: find  $\mathbf{p}$  minimizing  $\|\mathbf{A}\mathbf{p}\|^2$ 
  - If  $\text{rank}(\mathbf{A}) < 11$ , there are infinite solutions. Check if data is degenerate. Recalibrate.
  - If  $\text{rank}(\mathbf{A})$  is 11, solution given by SVD: eigenvector of  $\mathbf{A}$  with smallest eigenvalue
  - In practice, *the smallest eigenvalue of  $\mathbf{A}$  will not be exactly equal to zero, but will have a small value due to noise.*
  - Rule of thumb: Always check the smallest eigenvalue and/or the ratio between the largest and smallest values to estimate noise in the data.
  - High levels of noise mean error in the construction of the matrix  $\mathbf{A}$ .

# How many points do we need to fit the model?

$$\lambda \mathbf{x} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \mathbf{X}$$



Degrees of freedom  
(DOF)?

5

6

$$w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$\mathbf{M}$  is 3x4, so 12 unknowns, but projective scale ambiguity – 11 deg. freedom.

One equation per unknown -> **5 1/2 point correspondences determines a solution.**

**More than 5 1/2 point** correspondences -> overdetermined, many solutions to  $\mathbf{M}$ .

Least squares is finding the solution that best satisfies the overdetermined system.

*Why use more than 6? **Robustness** to error in feature points.*

# Camera calibration

- We can compute the matrix  $M$  from a set of point correspondences:
  - Given  $n$  ( $\geq 6$ ) correspondences  $x_i \rightarrow X_i$  (*pixels to world coordinates*)
  - Compute  $M = K [R \mid t]$  such that  $x_i = MX_i$  (or  $x_i = PX_i$ )
- But the **algorithm** for camera calibration has **two parts**
  1. *Compute the matrix  $M$  from a set of point correspondences*
  2. *Decompose  $M$  into *intrinsic and extrinsic* parameters, i.e.  $K$ ,  $R$  and  $t$*

# Camera calibration

- We can factorize  $M$  back to  $K [R \mid T]$
- We exploit the ***RQ factorization***  
(***R*** in ***RQ*** is not rotation matrix  $R$ ; crossed names!)
- ***R*** (upper triangular) is  $K$
- ***Q*** (orthogonal basis) is  $R$
- $t$ , the last column of  $[R \mid t]$ , is  
 $\text{inv}(K) * \text{last column of } M.$ 
  - But you need to do a bit of post-processing to make sure that the matrices are valid.
  - And, if the  $z$ -coordinates of the camera and world are pointing in the opposite direction, things can go wrong, e.g. in OpenGL the camera points in the negative  $z$  direction.

$$M = K [R \mid t]$$

# Limitations of the linear approach

- Using least square minimization to get '**M**' ( or '**P**') has little physical meaning.
- The method ignores constraints on the elements of '**P**'. The elements of '**P**' are not arbitrary, e.g. we may not be able to decompose it into an intrinsic and extrinsic parameter matrix.
- A more accurate approach is to use *constrained non-linear optimization* to find the calibration matrix.

# Constrained non-linear optimization

- Estimate  $P$  using one of the linear methods.  $P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}$
- Use this  $P$  as an initial guess and reproject the points on the image plane.
- Minimize the distance between all measured and reprojected image points.

$$\min_{\alpha, \beta, \gamma, u_0, v_0, R, t} \sum_{i=1}^N \left\{ \left( \frac{p_{11}X_i + p_{12}Y_i + p_{13}Z_i + p_{14}}{p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34}} - u_i \right)^2 + \left( \frac{p_{21}X_i + p_{22}Y_i + p_{23}Z_i + p_{24}}{p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34}} - v_i \right)^2 \right\}$$

- Ensure that  $R$  remains a rotation matrix
- Iterate until convergence



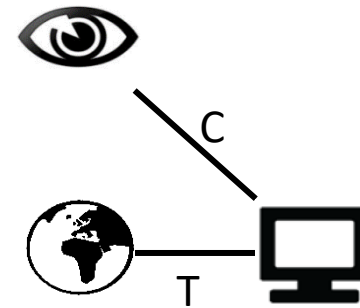
# Display calibration

# HMD calibration

- A **fully calibrated AR system** requires calibration not only of the input side, but also of the output side, namely, the **display**.
- With known internal and external **camera parameters**, we have enough information for presenting registered AR overlays on a video see-through display.
- For an optical see-through display, **head tracking** must be used instead of camera tracking to inform the registration of the AR overlay.
- The head tracking can be done, for example, with a camera attached to a head-mounted display. Nevertheless, head tracking alone does not identify **the pose of each eye relative to the display**.

# HMD calibration

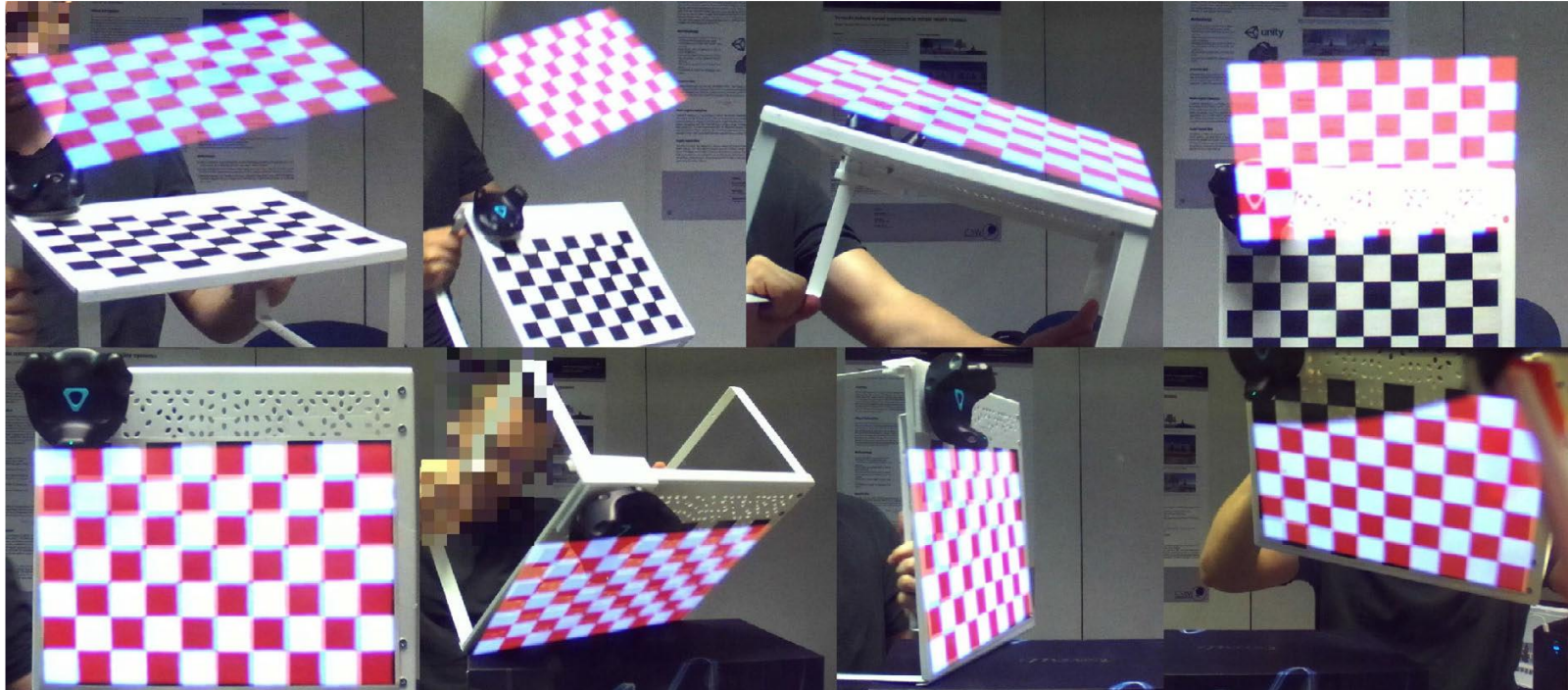
- We may assume that this transformation (eye-disply) is static and can be **calibrated** once the HMD is donned. This assumption will remain valid, unless major adjustments to the fitting of the HMD on the head are made during the session.



- *Because the user sees the composite image only in the optical see-through display, we must turn the usual image-based calibration approach around, **putting the human in the loop**.*
- The system displays calibration patterns, and **the user is asked to align** a structure in the physical environment with the pattern.

# HMD calibration

- Identifying the pose of each eye relative to the display is necessary.

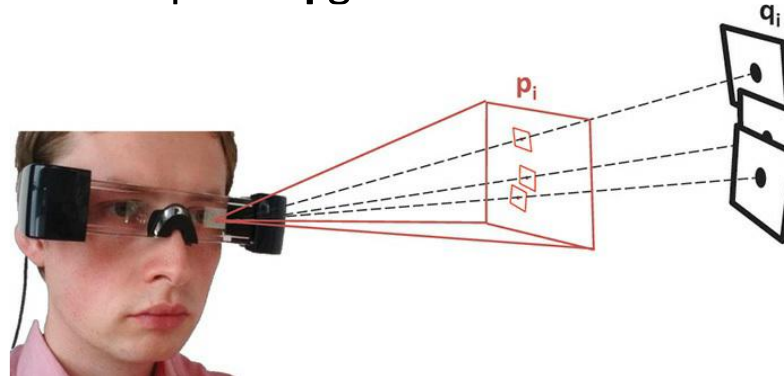


- The view from inside the OST HMD before (top row) and after (bottom row) the calibration.

Giorgio Ballestin, Manuela Chessa, and Fabio Solari. "A registration framework for the comparison of video and optical see-through devices in interactive augmented reality." *IEEE Access*, 9 (2021): 64828-64843.

# Single Point Active Alignment Method (SPAAM)

- The single point active alignment method (SPAAM) assumes that an optical see-through HMD is *tracked* relative to the world  $\mathbf{W}$ .
- The tracked point on the HMD is labeled  $\mathbf{H}$ , and the tracking transformation is denoted as  ${}^H\mathbf{M}_W$
- The eye  $\mathbf{E}$  of the user observes a point  $\mathbf{q}$  given in world coordinates at the 2D location  $\mathbf{p}$  in the display



- The single point active alignment method works by presenting a sequence of *crosshair targets* in the display, which *the user must align* with a known *real-world point*

Tuceryan, M., Genc, Y., and Navab, N. (2002) Single-point active alignment method (SPAAM) for optical see-through HMD calibration for augmented reality. Presence: Teleoperators and Virtual Environments 11, 3, MIT Press, 259–276

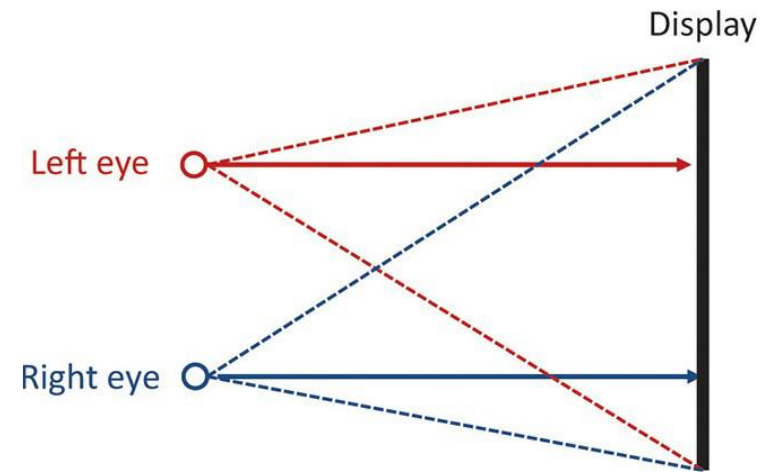
# Single Point Active Alignment Method (SPAAM)

- The goal of the calibration is to determine the projection matrix from head-to-eye coordinates  ${}^E\mathbf{M}_H$

$$\mathbf{p} = {}^E\mathbf{M}_H {}^H\mathbf{M}_W \mathbf{q}$$

- In general, *the display will not be exactly centered in front of the eye*, which leads to an off-axis projection (i.e., the internal camera parameters  $p_x$  and  $p_y$  will not correspond to the screen center)

- The resulting viewing frustum will be asymmetric



# Single Point Active Alignment Method (SPAAM)

- The goal of SPAAM is to compute  ${}^E\mathbf{M}_H$  from at least *six* 2D–3D correspondences (or, better yet, 12–20 correspondences) obtained with user interaction.
- A series of *crosshair markers* displayed at locations  $\mathbf{p}_i$  on the screen are presented for the user. The user must visually align the crosshair with a known world point  $\mathbf{q}$
- When the user confirms the alignment by pressing a trigger button, the system records the 2D–3D correspondence  $(\mathbf{p}_i, \mathbf{q}_i)$  with  $\mathbf{q}_i = {}^H\mathbf{M}_w \mathbf{q}$  and advances to the next calibration point.
- The desired projection matrix can be computed from these correspondences  $\mathbf{p}_i = {}^E\mathbf{M}_H \mathbf{q}_i$  using the *direct linear technique* described in the camera calibration slides.

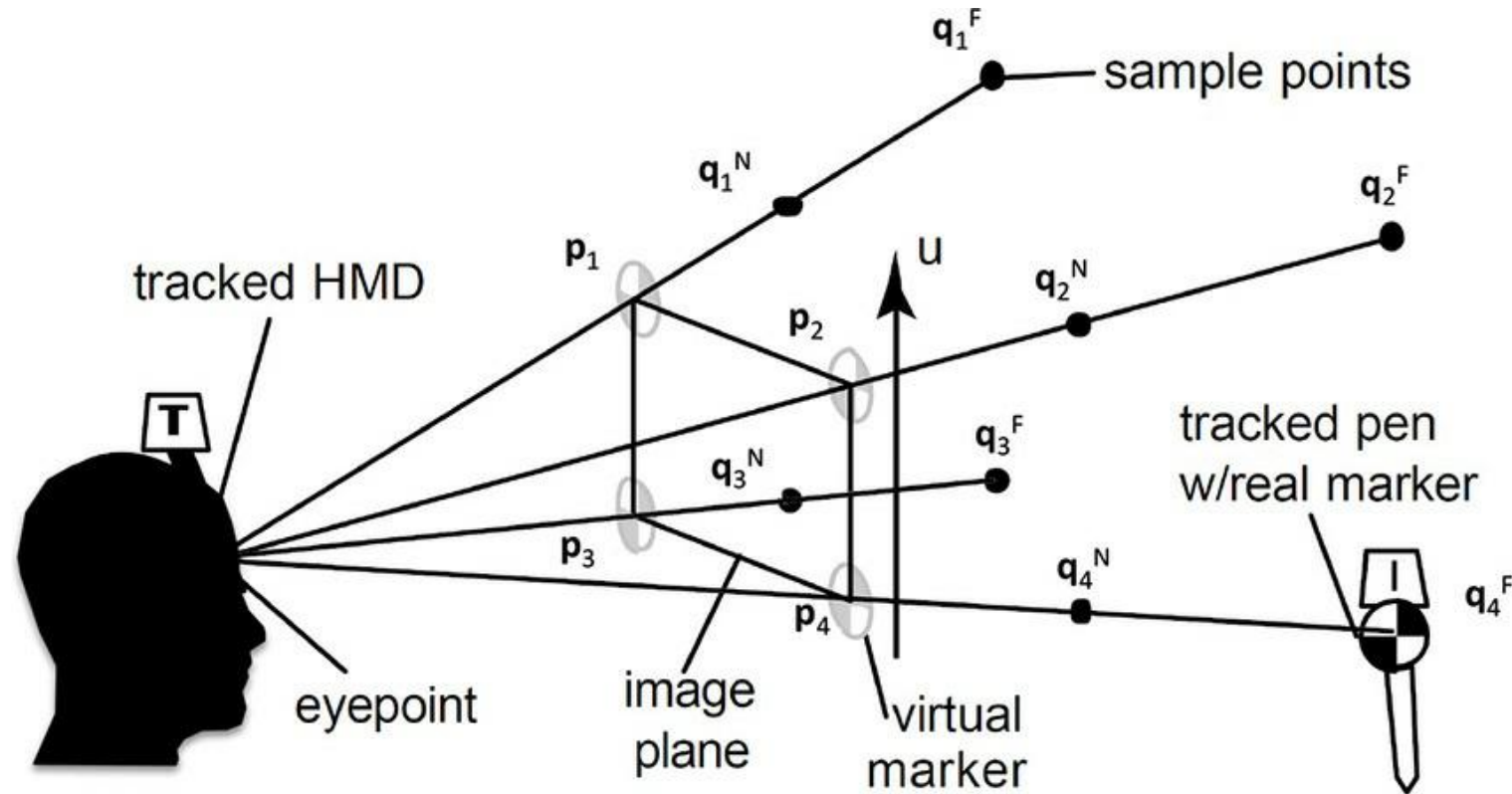
# HMD Calibration Using a Pointing Device

- This calibration method requires an additional **tracked pointing device**, which must be aligned with the on-display crosshair instead of the static calibration point used by SPAAM.
- Such a pointing device will often be part of the AR setup, including a trigger to confirm the alignment.
- The advantage of the pointing device is that instead of having to move the head to achieve the alignment, the user can move the arm, which is typically both more precise and more convenient.

Fuhrmann, A., Schmalstieg, D., and Purgathofer, W. (2000) Practical calibration procedures for augmented reality. Proceedings of the Eurographics Workshop on Virtual Environments (EGVE), 3–12.



# HMD Calibration Using a Pointing Device



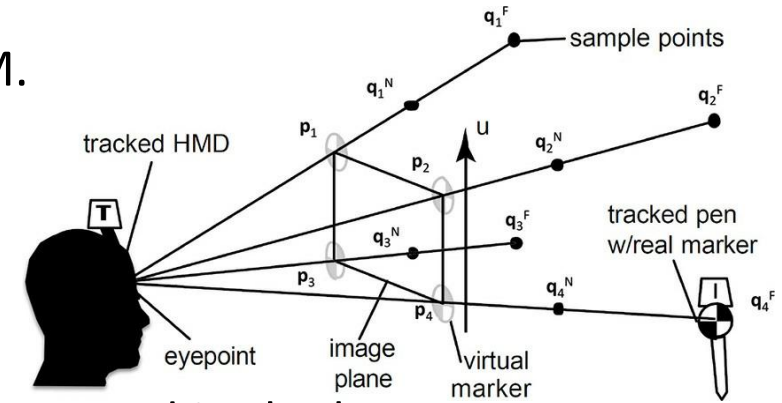
- The pointing device can replace the fixed known point in the world used in SPAAM with direct user input. Here, the user can manually select the distance of the 3D points by stretching out the arm.

# HMD Calibration Using a Pointing Device

- This method requires the user **to define lines** rather than individual points, by specifying **two points** for the **same crosshair** location: one near location  $\mathbf{q}^N$ , with the arm held close to the head, and one far location  $\mathbf{q}^F$ , with the arm fully stretched out.
- The procedure is repeated four times, giving one line  $(\mathbf{q}_i^N, \mathbf{q}_i^F)$  per crosshair point  $\mathbf{p}_i$  near a corner of the screen, thereby approximating the viewing frustum.
- If the transformation of the pointing device  $\mathbf{D}$  is tracked as  ${}^D\mathbf{M}_w$ , we store  ${}^H\mathbf{M}_w ({}^D\mathbf{M}_w)^{-1}$  for each of the eight input points.

# HMD Calibration Using a Pointing Device

- Recovery of the transformation from  $\mathbf{H}$  to  $\mathbf{E}$  can be based on geometric considerations, or it could use a *direct linear method* equivalent to the one in SPAAM.



- The **geometric method** may be faster to compute.
- First, the four lines should intersect in the eyepoint, which is computed in the least squares sense. Averaging the direction of the lines yields a reasonable approximation of the viewing direction, which is initially assumed to be orthogonal to the image plane.
- The vertical and horizontal directions of the image plane are estimated from the intersections of the image plane with the lines.
- These estimates are often sufficient, if followed by nonlinear refinement.



**Università  
di Genova**

**DIBRIS** DIPARTIMENTO  
DI INFORMATICA, BIOINGEGNERIA,  
ROBOTICA E INGEGNERIA DEI SISTEMI

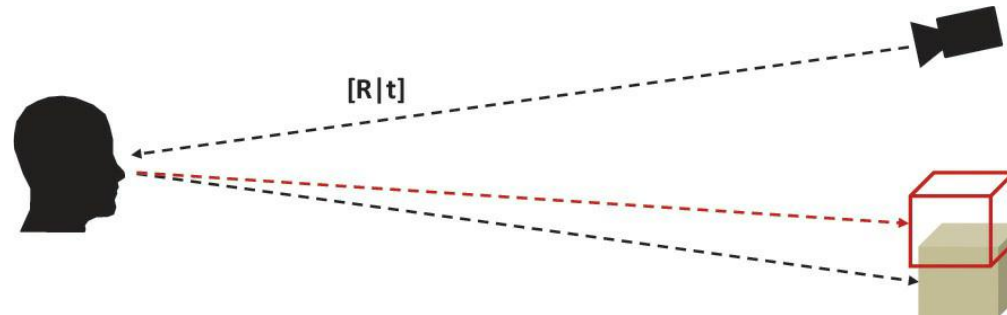
# Issues and error sources in AR registration

# Registration

- The complex interplay of AR system components implies that many potential sources of **error** can affect **registration** (*allowing the overlay of AR and real contents*).
- We can distinguish between
  - **static error**, which affects *accuracy*, and
  - **dynamic error**, which affects *precision*.
- Correction of **static error** mainly requires improved **calibration**.
- *Dynamic errors* are most severe, as they *cannot be addressed with static calibration*. In this category, we mainly must combat error propagation and latency.

# Error Propagation

- A problem that plagues many practical AR systems is **error propagation**, which can **amplify small errors** resulting from *jittery tracking* or *insufficient calibration*.
- While the original error might be small enough to pass unnoticed, the amplified, dependent error may no longer be tolerable.
- The most common problem is that **small rotational errors** will lead to **large translational errors**.



- The effects of error propagation can be minimized *by avoiding the dynamic concatenation* of coordinate systems in favor of directly expressing the relationship of one coordinate system in terms of the other.

# Latency

- If user motion is observed by a tracking system, a corresponding image cannot be presented immediately.
- The ***end-to-end-delay*** consists of several components:
  - The physical measurement process performed by the sensor and the time to transmit the result to the host computer
  - The processing of the measurement by the host computer
  - The image generation performed by the host computer
  - Video synchronization between the image generator and the display
  - An internal delay in the display until the image is finally shown
- For a moving user, the **temporal error** resulting from latency directly translates **into spatial error**, because the augmented images are presented at a wrong (outdated) position or have a wrong (outdated) camera pose.

# Filtering and Prediction

- If measurements suffer from jitter, the **sensor data** must be **filtered** to become smoother.
- With filter-induced noise reduction, we can use a suitable motion model to **predict** and, therefore, compensate for a certain amount of latency.
- Widely used approaches for statistical filtering of sensor data include the Kalman filter and the particle filter. Both can be formulated as “recursive” filters, which rely on the most recently computed state, thereby running in a tracking loop with a constant memory requirement.
- The **Kalman filter** assumes that the error can be described by a normal distribution and that a linear combination of the system state and the measurement can be used to identify and remove the error.
- Most practical sensor systems have nonlinear behavior, which can be addressed with more advanced models such as the **extended Kalman filter** and the unscented transform.
- A **particle filter** can be used if the error cannot be approximated with a normal distribution