Module IN 2018

Introduction to Augmented Reality

Prof. Gudrun Klinker

Camera Calibration

SS 2018

1. ...

Literature

Roger Y. Tsai: A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. IEEE J. Rob. Autom. 3(4): 323-344 (1987).

http://www.vision.caltech.edu/bouguetj/calib_doc/papers/Tsai.pdf

Agenda

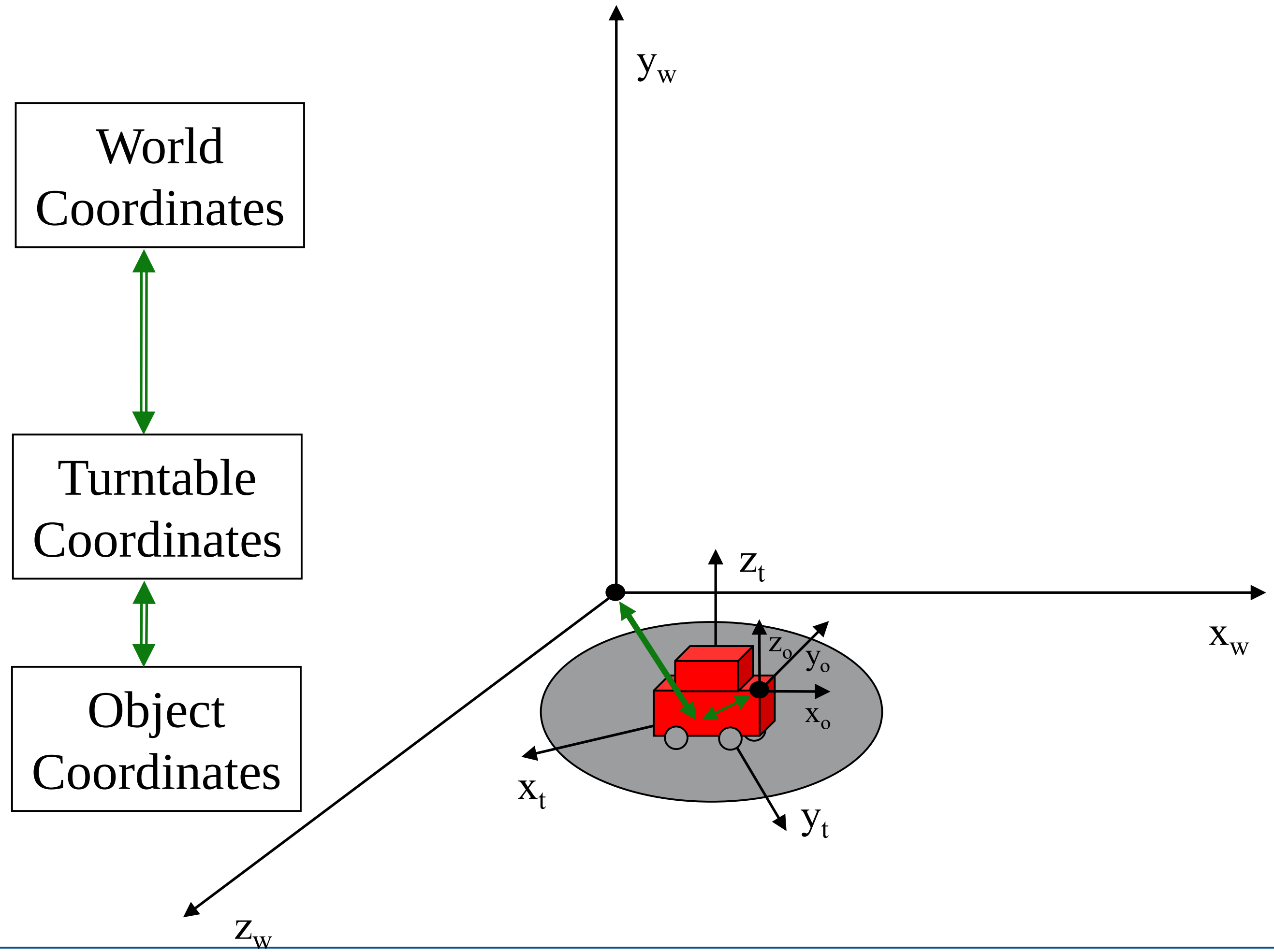
1. Overview
2. Image Formation Steps
3. Two-Stage Camera Calibration
4. How to use Tsai-code with OpenGL

1. Overview

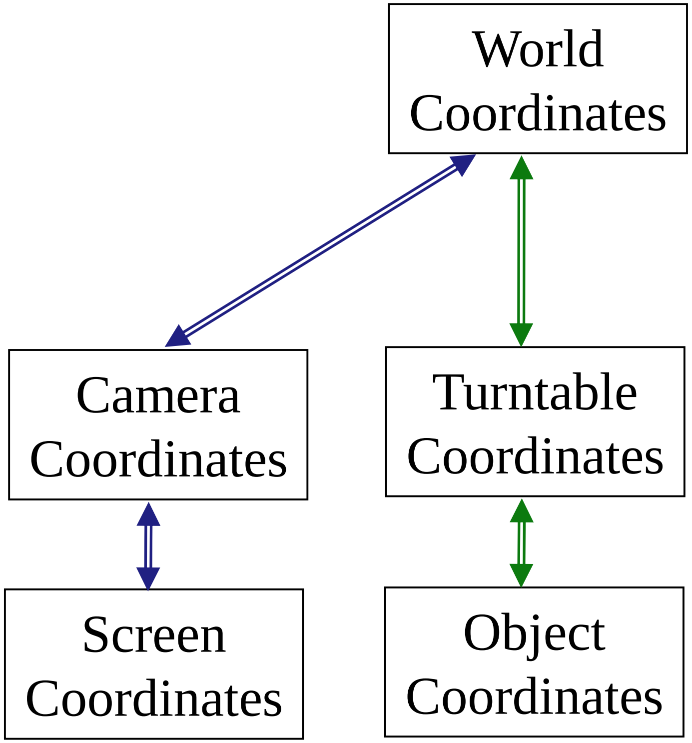
1.1 Reminder: Coordinate Systems

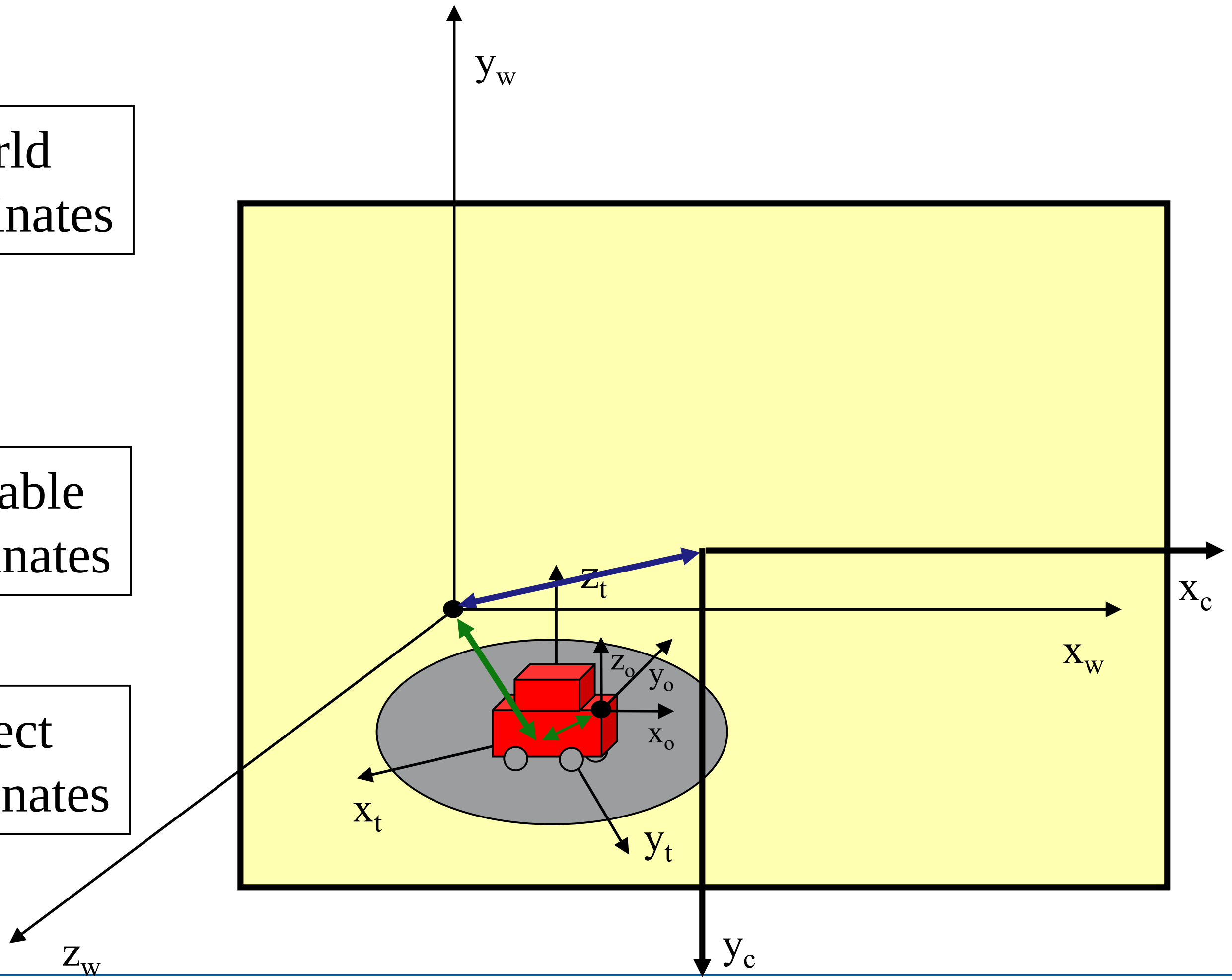
1.2 Tsai‘s Camera Calibration

1.1 Reminder: Coordinate Systems

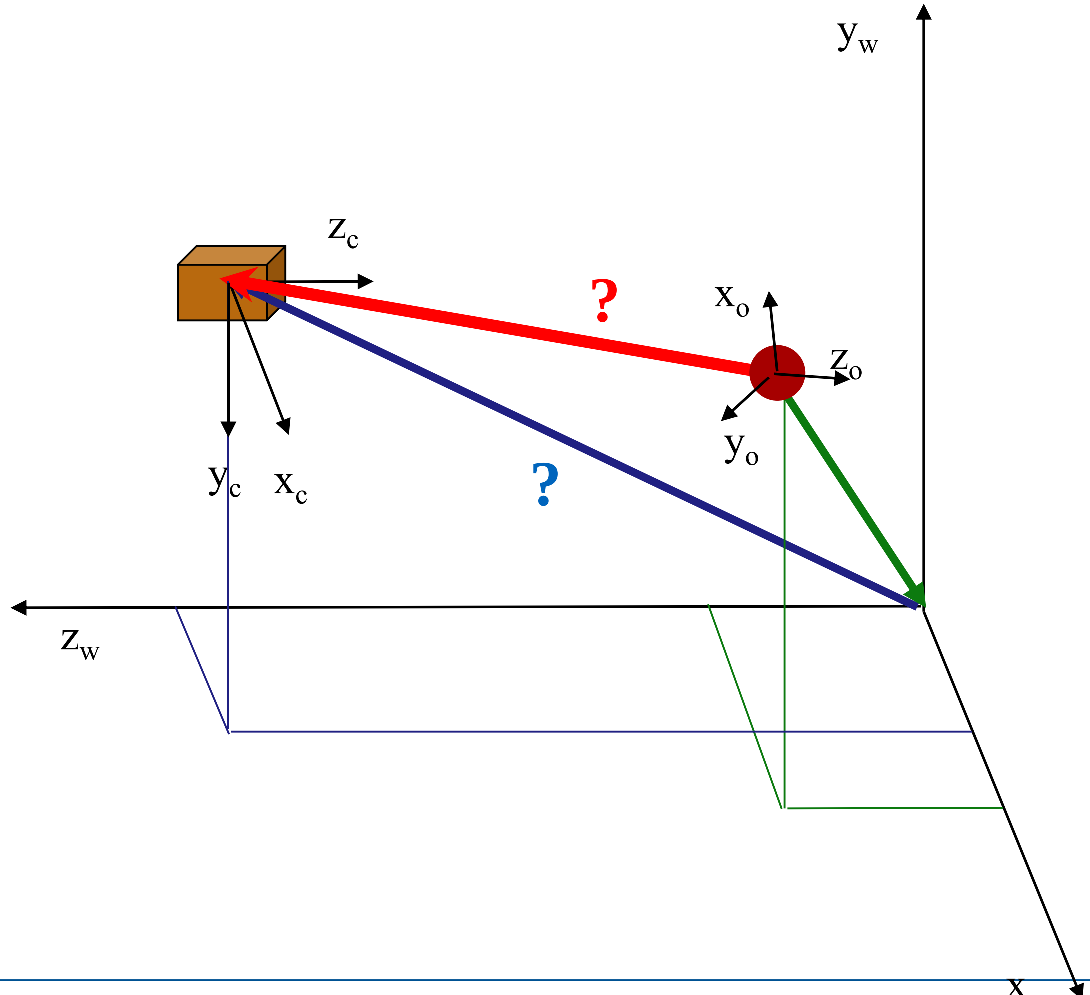


1.1 Reminder: Coordinate Systems

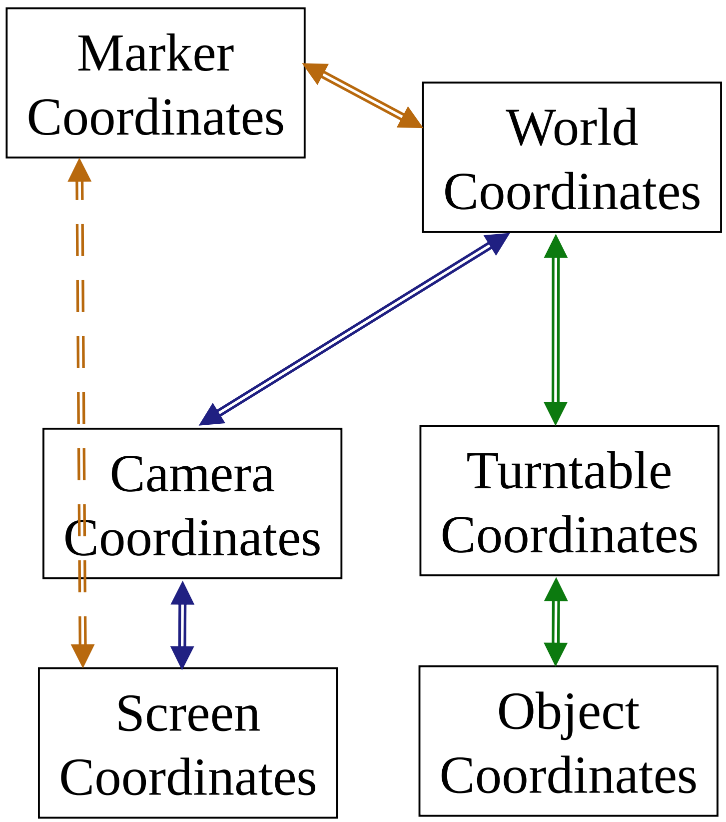


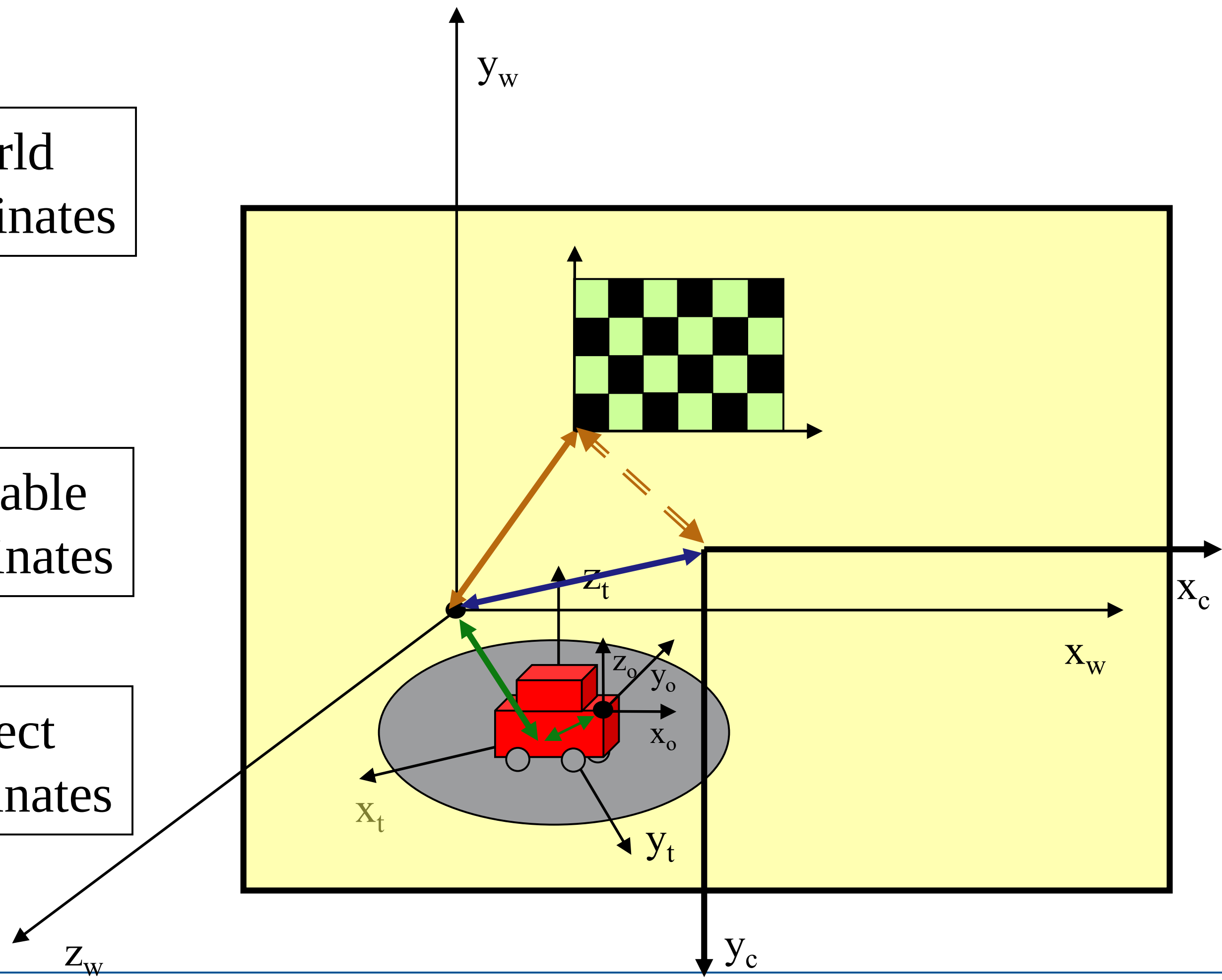


1.1 Reminder: Coordinate Systems



1.1 Reminder: Coordinate Systems





1. Overview

1.1 Reminder: Coordinate Systems

→ 1.2 Tsai‘s Camera Calibration

1.2 Tsai‘s Camera Calibration

- Extrinsic parameters (“pose estimation”)
- Intrinsic parameters (“calibration”)

- Tsai’s Approach:
  - Model of image formation steps
  - Two-stage algorithm to determine model parameters

1.2 Tsai‘s Camera Calibration

Tsai’s Approach:

- World-to-camera transformation: rotation followed by translation
- 6 intrinsic and 6 extrinsic camera parameters
- Parallelism constraint under radial distortion, variable focal length and translation in z
- 2-stage algorithm:
  - Compute 3D orientation, T_x and T_y
  - Compute focal length, distortion coefficients and T_z
  - (other parameters taken from device specifications)
- Separate solutions for coplanar and non-coplanar sets of markers

Agenda

1. Overview
2. Image Formation Steps
3. Two-Stage Camera Calibration
4. How to use Tsai-code with OpenGL

2. Image Formation Steps

2.1 Overview

Given: Points in world coordinates (x_w, y_w, z_w)

R, T — World-to-camera transformation.

(x, y, z) Camera coordinates

f — Idealized 3D-to-2D projection.

(X_u, Y_u) Ideal image coordinates

κ_1, κ_2 — Radial lens distortion.

(X_d, Y_d) True image coordinates

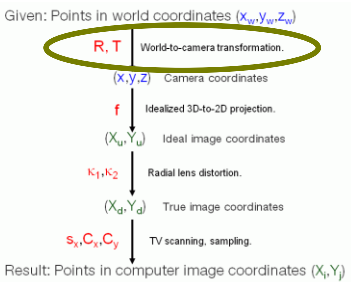
s_x, C_x, C_y — TV scanning, sampling.

Result: Points in computer image coordinates (X_i, Y_j)

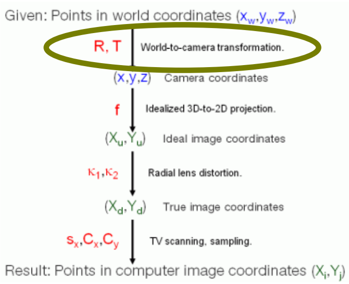
Pose estimation (6 extrinsic parameters, 6 DoF)

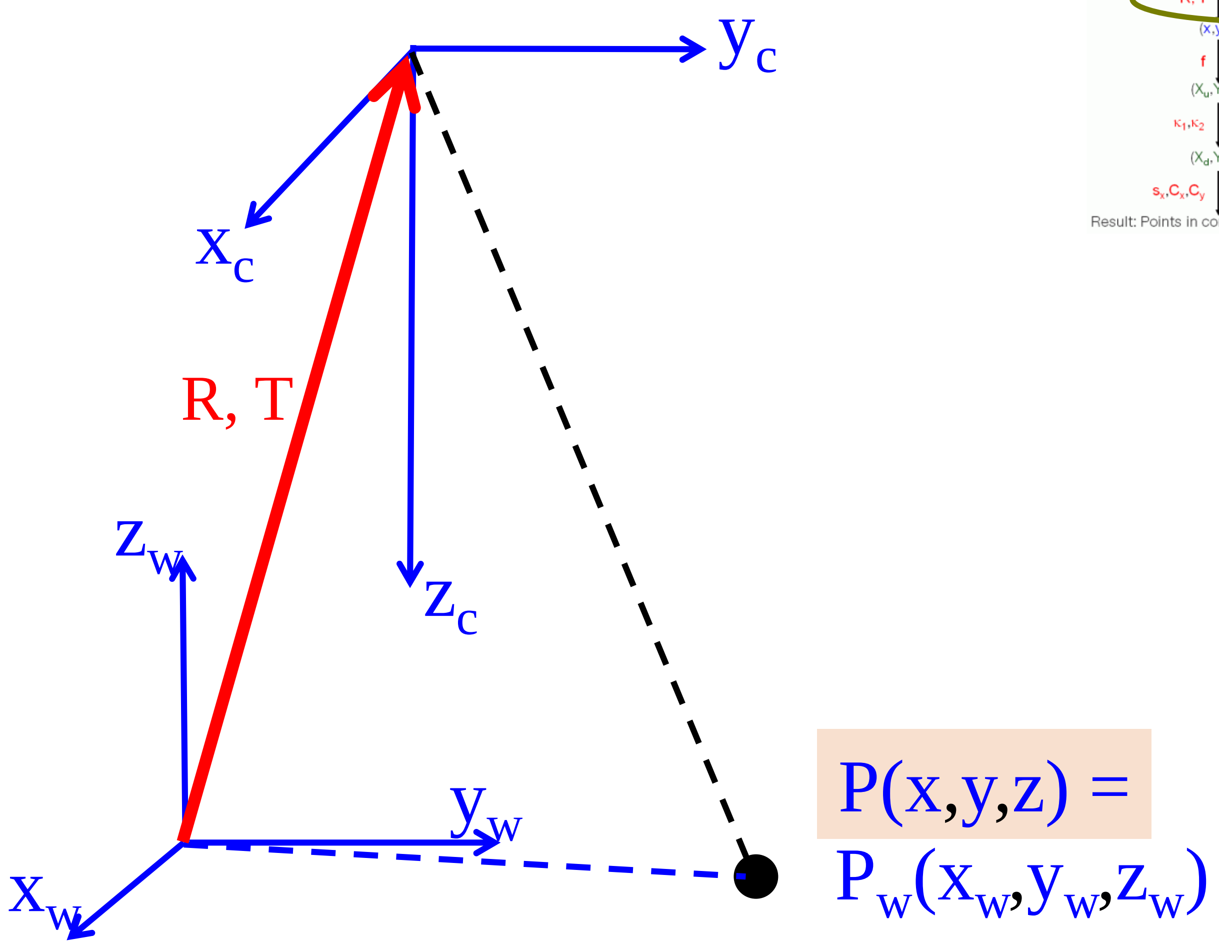
Camera calibration (6 intrinsic parameters)

2. Image Formation Steps



2.2 Step 1: World-to-Camera Transformation





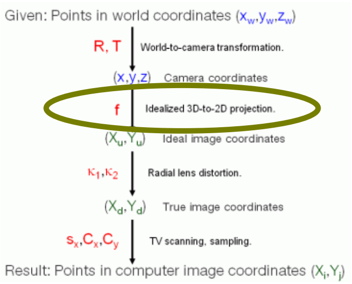
2.2 Step 1: World-to-Camera Transformation

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = R \begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} + T$$

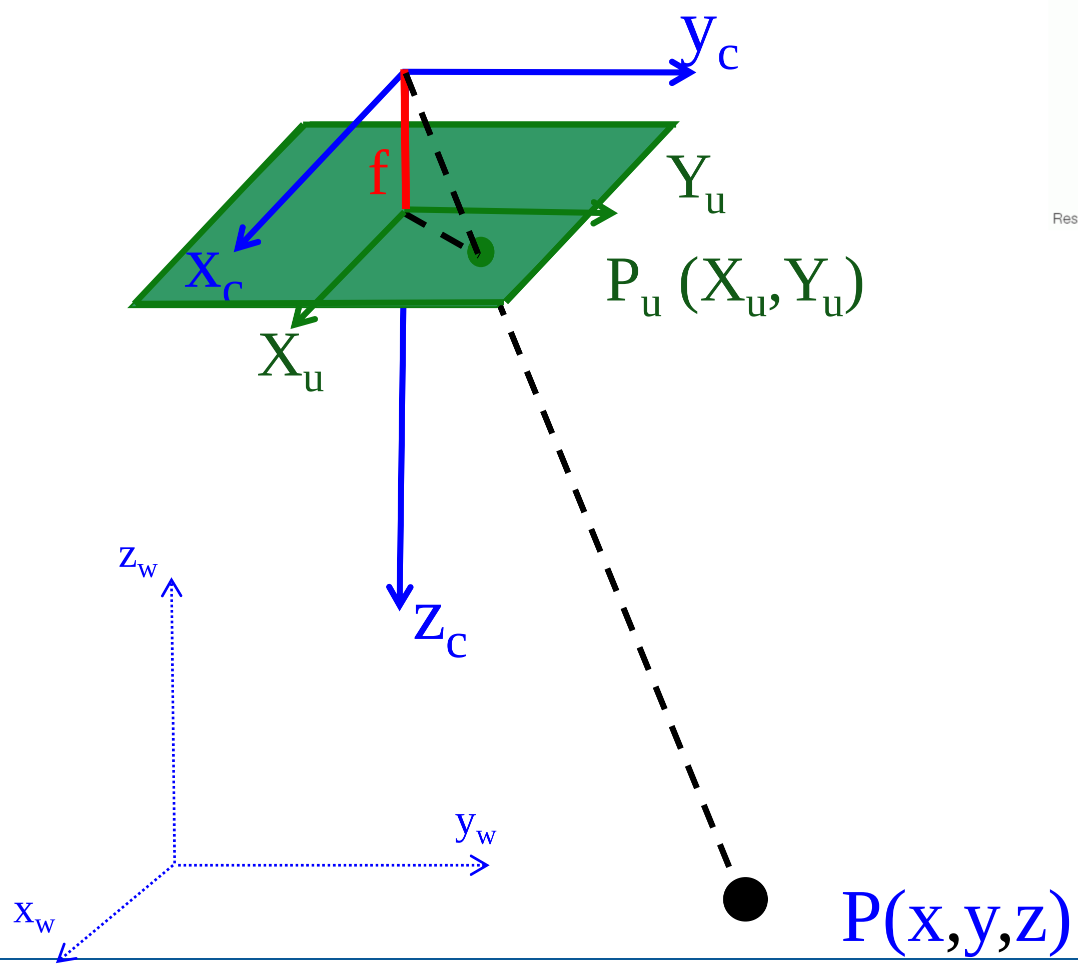
$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{pmatrix} \begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} + \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix}$$

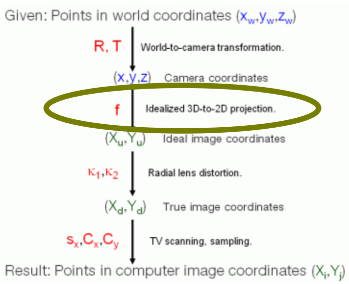
$$\begin{aligned} x &= r_1 x_w + r_2 y_w + r_3 z_w + T_x \\ y &= r_4 x_w + r_5 y_w + r_6 z_w + T_y \\ z &= r_7 x_w + r_8 y_w + r_9 z_w + T_z \end{aligned}$$

2. Image Formation Steps



2.3 Step 2: Idealized 3D-to-2D Projection





2.3 Step 2: Idealized 3D-to-2D Projection

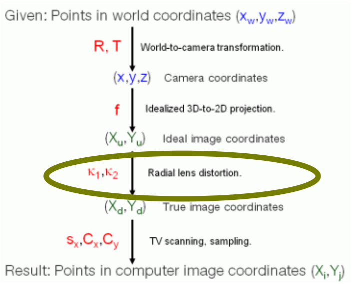
$$X_u = f \ x/z$$

$$= f \frac{r_1 x_w + r_2 y_w + r_3 z_w + T_x}{r_7 x_w + r_8 y_w + r_9 z_w + T_z}$$

$$Y_u = f \ y/z$$

$$= f \frac{r_4 x_w + r_5 y_w + r_6 z_w + T_y}{r_7 x_w + r_8 y_w + r_9 z_w + T_z}$$

2. Image Formation Steps



2.1 Overview

2.2 Step 1: World-to-Camera Transformation

2.3 Step 2: Idealized 3D-to-2D Projection

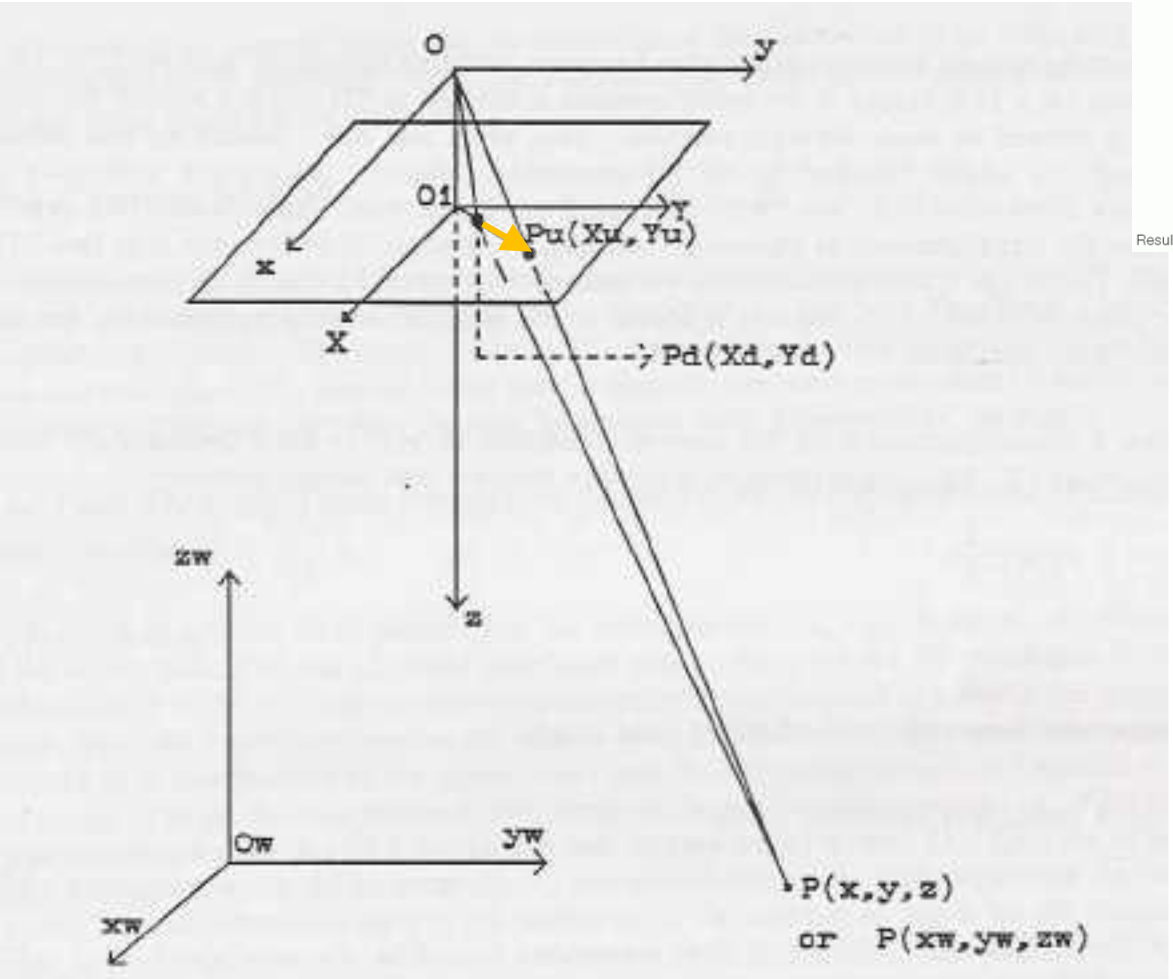
→ 2.4 Step 3: Radial Lens Distortion

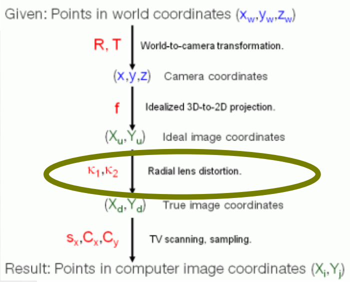
2.5 TV Scanning, Sampling

2.6 Summary

2. Image Formatin Steps

2.4 Step 3: Radial Lens Distortion





2.4 Step 3: Radial Lens Distortion

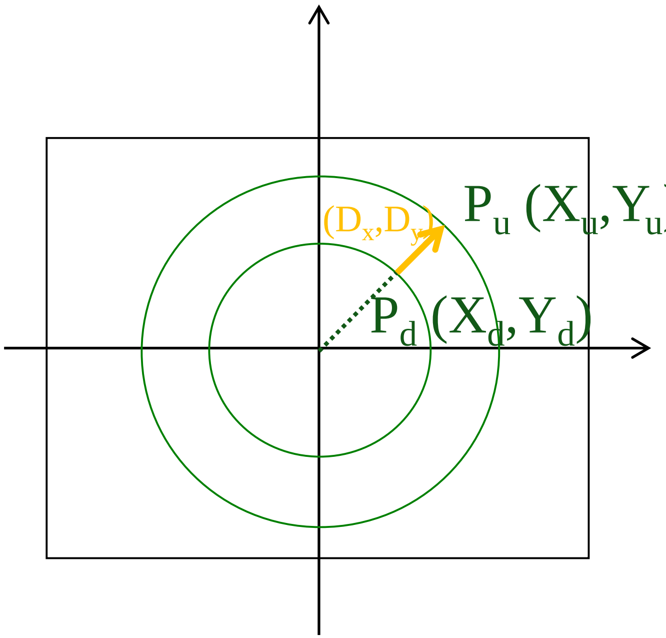
$$X_d = X_u - D_x$$
$$Y_d = Y_u - D_y$$

$$\boxed{X_u = X_d + D_x \quad Y_u = Y_d + D_y}$$

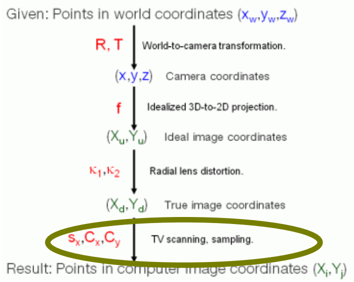
with:

$$D_x = X_d\,(\kappa_1 r^2 + \kappa_2 r^4)$$
$$D_y = Y_d\,(\kappa_1 r^2 + \kappa_2 r^4)$$

$$r = \sqrt{X_d^2 + Y_d^2}$$



2. Image Formation Steps



2.1 Overview

2.2 Step 1: World-to-Camera Transformation

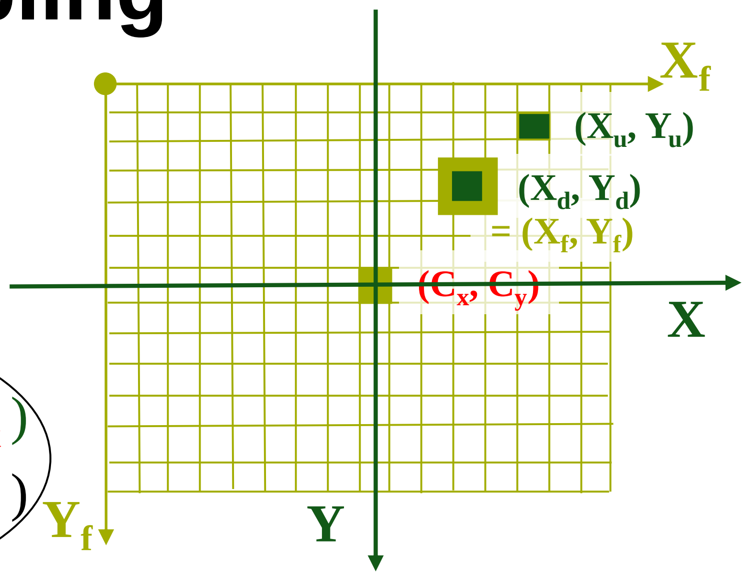
2.3 Step 2: Idealized 3D-to-2D Projection

2.4 Step 3: Radial Lens Distortion

→ 2.5 TV Scanning, Sampling

2.6 Summary

2.5 TV Scanning, Sampling



$$X_f = s_x X_d / d_x' + C_x \qquad Y_f = Y_d / d_y + C_y$$

$$X_d = d_x'/s_x \,(X_f - C_x) \qquad Y_d = d_y \,(Y_f - C_y)$$

with: (X_f, Y_f): row, column of image pixel in frame memory

s_x: scale factor ("aspect ratio")

(C_x, C_y): center of computer image

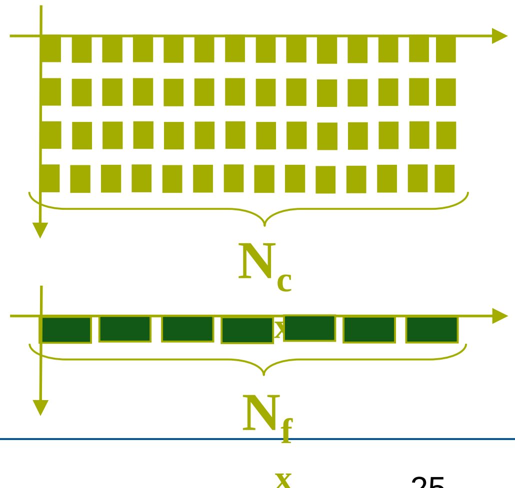
d_x: distance between adjacent sensor elements (x-dir)

d_y: distance between adjacent sensor elements (y-dir)

$d_x' = d_x \,(N_{cx} / N_{fx})$

N_{cx}: number of sensor elements in x-direction

N_{fx}: number of pixels in a line ("image width")



2. Image Formation Steps

2.6 Summary

$$X_u = f\ x/z$$

$$X_u = X_d + D_x = X_d + X_d(\kappa_1 r^2 + \kappa_2 r^4)$$

$$f\,\frac{r_1 x_w + r_2 y_w + r_3 z_w + T_x}{r_7 x_w + r_8 y_w + r_9 z_w + T_z}$$

$$= (d_x{}' / s_x)(Xf\_Cx)[1 + (\kappa_1 r^2 + \kappa_2 r^4)]$$

$$Y_u = f\ y/z$$

$$Y_u = Y_d + D_y = Y_d + Y_d(\kappa_1 r^2 + \kappa_2 r^4)$$

$$f\,\frac{r_4 x_w + r_5 y_w + r_6 z_w + T_y}{r_7 x_w + r_8 y_w + r_9 z_w + T_z}$$

$$= d_y(Yf\_Cy)[1 + (\kappa_1 r^2 + \kappa_2 r^4)]$$

Agenda

1. Overview
2. Image Formation Steps
3. Two-Stage Camera Calibration
4. How to use Tsai-code with OpenGL

3. Two-Stage Camera Calibration

3.1 Overview

- Preparation:
  - Determine N_{cx}, N_{fx}, $d_x{}'$ and d_y from device specifications, assume (C_x, C_y) is the center pixel of the image
  - Measure markers (i:1..N) in the scene $(x_w, y_w, z_w)_i$
  - For current image: determine computer image coordinates $(X_f, Y_f)_j$ of all visible markers ($j \varepsilon$ 1..N)
- Stage 1:
  Compute 3D orientation (R) and position (T_x, T_y) and scale factor (s_x)
- Stage 2:
  Compute effective focal length (f) distortion coefficients (κ_1, κ_2), and z position T_z

3. Two-Stage Camera Calibration

3.2 Stage 1: Compute 3D Orientation and Position

- Get distorted image coordinates (X_d, Y_d)
- Compute transformation parameters (5-7 mixed terms of R and T)
- Determine R and T from mixed terms

3.2 Stage 1: Compute 3D Orientation and Position

Illustration

$$X_u = f\ x/z \qquad\qquad \boxed{f\ \frac{r_1 x_w + r_2 y_w + r_3 z_w + T_x}{r_7 x_w + r_8 y_w + r_9 z_w + T_z}}$$

$$X_u = X_d + D_x = X_d + X_d(\kappa_1 r^2 + \kappa_2 r^4) \qquad \boxed{= (d_x' / s_x)(X_f - C_x)\,[1 + (\kappa_1 r^2 + \kappa_2 r^4)]}$$

$$Y_u = f\ y/z \qquad\qquad \boxed{f\ \frac{r_4 x_w + r_5 y_w + r_6 z_w + T_y}{r_7 x_w + r_8 y_w + r_9 z_w + T_z}}$$

$$Y_u = Y_d + D_y = Y_d + Y_d(\kappa_1 r^2 + \kappa_2 r^4) \qquad \boxed{= d_y(Y_f - C_y)\,[1 + (\kappa_1 r^2 + \kappa_2 r^4)]}$$

- 12 unknowns
- 2 formulas per matching point

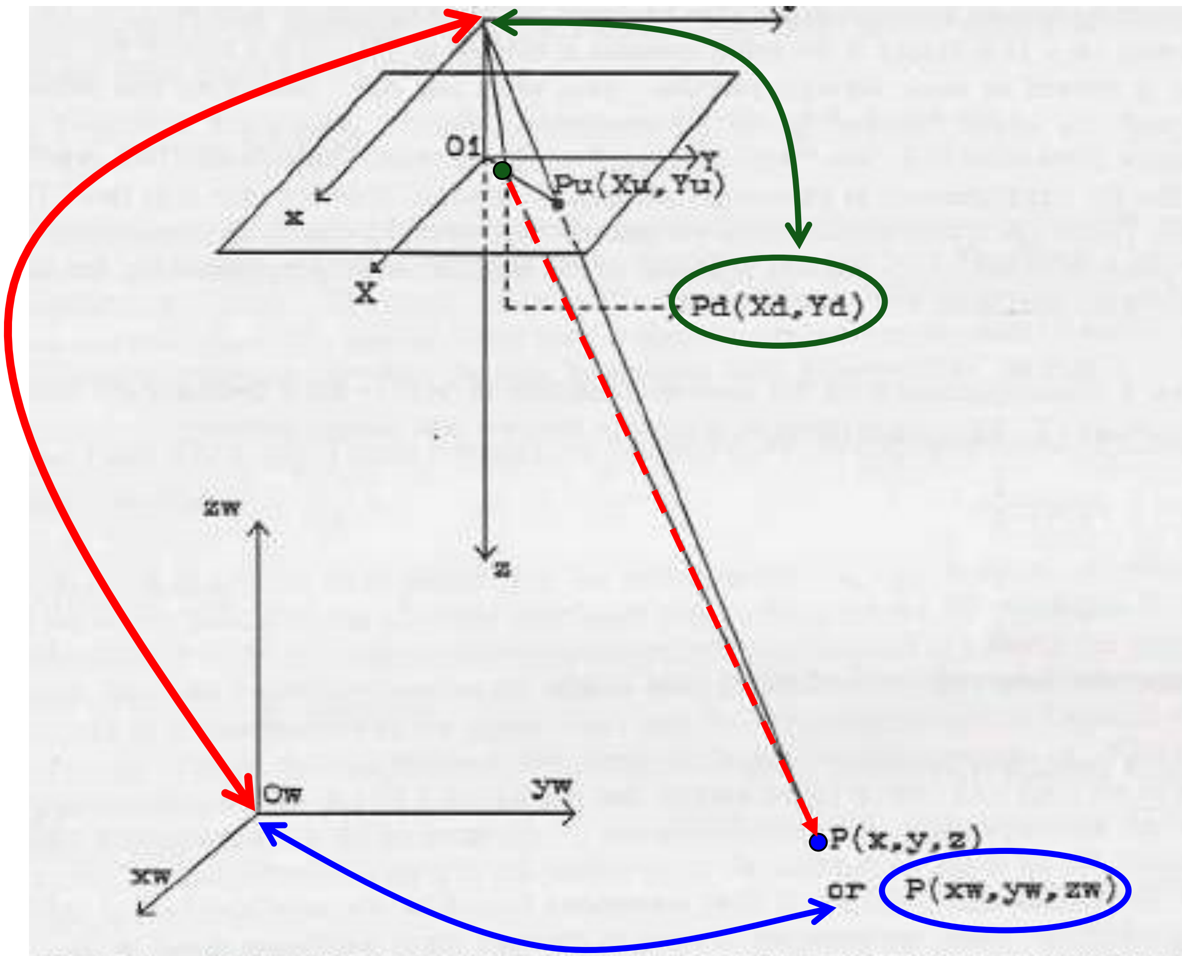


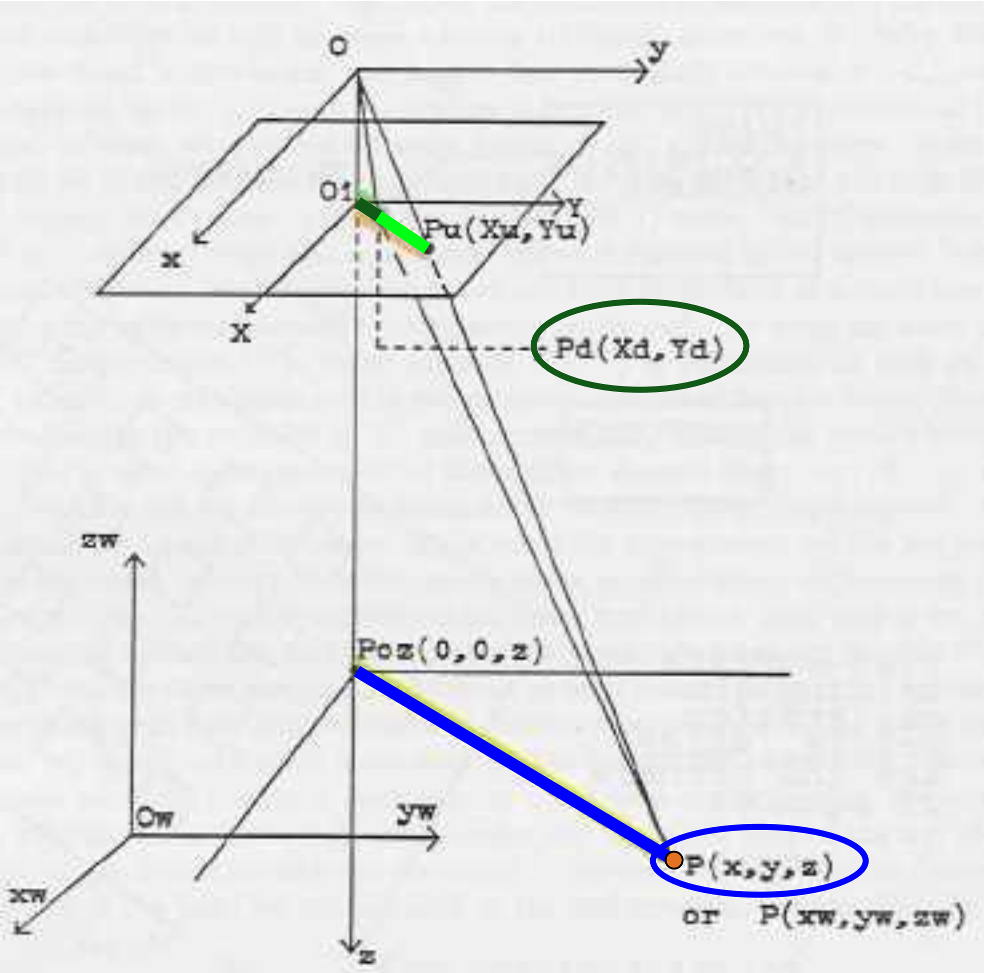
Figure 1. Camera Geometry with Perspective Projection and Radial Lens Distortion

3.2 Stage 1: Compute 3D Orientation and Position

Parallelism Constraint

Radial distortion does not alter the direction of the vector from the image center 0_1 (i.e. origin O_f) to the image point (P_u or P_d, rsp):

$O_f P_u \parallel O_f P_d \parallel P_{oz} P$



3.2 Stage 1: Compute 3D Orientation and Position

Parallelism Constraint

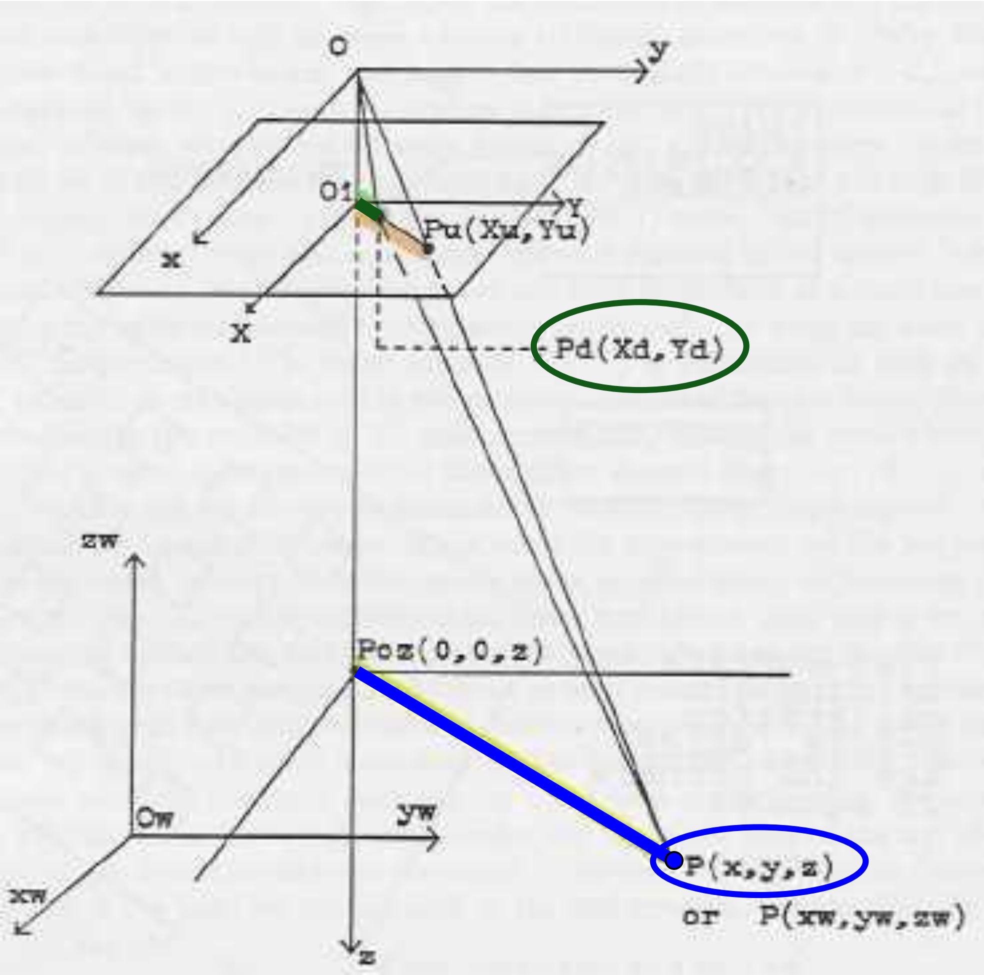
Radial distortion does not alter the direction of the vector from the origin to the image point:

$$\overline{O_f\,P_d} \parallel \overline{P_{oz}\,P}$$

$$\overline{\begin{pmatrix}0\\0\\f\end{pmatrix}\begin{pmatrix}X_d\\Y_d\\f\end{pmatrix}} \parallel \overline{\begin{pmatrix}0\\0\\z\end{pmatrix}\begin{pmatrix}x\\y\\z\end{pmatrix}}$$

$$\begin{pmatrix}X_d\\Y_d\end{pmatrix} = \alpha \begin{pmatrix}x\\y\end{pmatrix}$$

$$X_d y = Y_d x$$



3.2 Stage 1: Compute 3D Orientation and Position

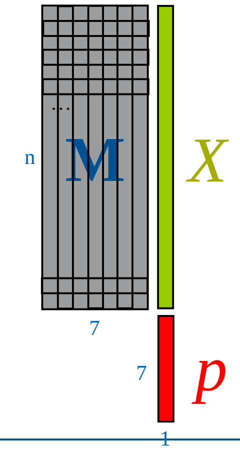
Parallelism Constraint

$$
\begin{aligned}
X_d\, y &= Y_d\, x \\
X_d\,(r_4\, x_w + r_5\, y_w + r_6\, z_w + T_y) &= Y_d\,(r_1\, x_w + r_2\, y_w + r_3\, z_w + T_x) \\
X_d\, T_y &= Y_d\,(r_1\, x_w + r_2\, y_w + r_3\, z_w + T_x) \\
&\quad - X_d\,(r_4\, x_w + r_5\, y_w + r_6\, z_w)
\end{aligned}
$$

$$
\begin{aligned}
X_d = {} & Y_d\, x_w\,(r_1/T_y) + Y_d\, y_w (r_2/T_y) + Y_d\, z_w (r_3/T_y) + Y_d\,(T_x/T_y) \\
& - X_d\, x_w\,(r_4/T_y) - X_d\, y_w\,(r_5/T_y) - X_d\, z_w\,(r_6/T_y)
\end{aligned}
$$

Written as vectors and matrix for all markers (i: 1..N):

$$
\begin{aligned}
X_{di} = {} & [Y_{di}\, x_{wi} \quad Y_{di}\, y_{wi} \quad Y_{di}\, z_{wi} \quad Y_{di} \quad -X_{di}\, x_{wi} \quad -X_{di}\, y_{wi} \quad -X_{di}\, z_{wi}] \circ \\
& [r_1/T_y \quad r_2/T_y \quad r_3/T_y \quad T_x/T_y \quad r_4/T_y \quad r_5/T_y \quad r_6/T_y]
\end{aligned}
$$



Data from each marker becomes

- a value in a 1xN result vector X and
- a row in an Nx7 matrix, M,
- to be multiplied with the 1x7 parameter vector p: $X = M\, p$

3.2 Stage 1: Compute 3D Orientation and Position

Computation of Transformation Parameters (Mixed Terms of R and T)

Two cases for using the parallelism constraint:

- Coplanar markers: $z_w = 0$

  $$X_d = [Y_d x_w \quad Y_d y_w \quad \cancel{Y_d z_w} \quad Y_d \quad -X_d x_w \quad -X_d y_w \quad \cancel{-X_d z_w}]_i \circ [r_1 / T_y \quad r_2 / T_y \quad \cancel{r_3 / T_y} \quad T_x / T_y \quad r_4 / T_y \quad r_5 / T_y \quad \cancel{r_6 / T_y}]$$

  System of N linear equations with 5 parameters

- Non-coplanar markers
  System of N linear equations with 7 parameters

The Tsai-code uses a routine (lmdif_ from the MINPACK library) that minimizes the square error between predicted and actual marker locations in the image (based on the Levenberg-Marquardt algorithm)

3.2 Stage 1: Compute 3D Orientation and Position

Computation of Transformation Parameters (Mixed Terms of R and T)

Coplanar Case

5 Parameters: r_1/T_y, r_2/T_y, T_x/T_y, r_4/T_y, r_5/T_y

$$R: \begin{pmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{pmatrix} \quad \text{submatrix } C: \begin{pmatrix} r_1' & r_2' & . \\ r_4' & r_5' & . \\ . & . & . \end{pmatrix} = \begin{pmatrix} r_1/T_y & r_2/T_y & . \\ r_4/T_y & r_5/T_y & . \\ . & . & . \end{pmatrix}$$

Observations:

- Unit row and column vectors,
- Mutually orthogonal column vectors

$$T_y^2 = \frac{S_r \pm \sqrt{S_r^2 - 4(r_1' r_5' - r_4' r_2')}}{2\,(r_1' r_5' - r_4' r_2')^2}, \qquad \text{with: } S_r = r_1'^2 + r_2'^2 + r_4'^2 + r_5'^2$$

Determine sign of T_y: try out +/- 3D-to-2D projection for a non-central marker

3.2 Stage 1: Compute 3D Orientation and Position

Computation of Transformation Parameters (Mixed Terms of R and T)

Coplanar Case

$$r_1 = r_1' \, T_y$$
$$r_2 = r_2' \, T_y$$
$$r_4 = r_4' \, T_y$$
$$r_5 = r_5' \, T_y$$
$$T_x = T_x' \, T_y$$

$$R = \begin{pmatrix} r_1 & r_2 & \sqrt{1 - r_1^2 - r_2^2} \\ r_4 & r_5 & s\sqrt{1 - r_4^2 - r_5^2} \\ r_7 & r_8 & r_9 \end{pmatrix}$$

$$(r_7 \quad r_8 \quad r_9) = (r_1 \quad r_2 \quad r_3) \times (r_4 \quad r_5 \quad r_6)$$

$$s = -\,\mathrm{sgn}\,(r_1 r_4 + r_2 r_5)$$

3.2 Stage 1: Compute 3D Orientation and Position

Computation of Transformation Parameters (Mixed Terms of R and T)

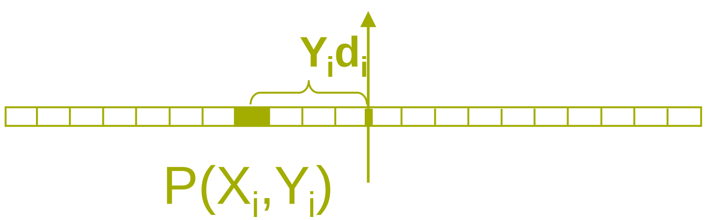
Non-Coplanar Case

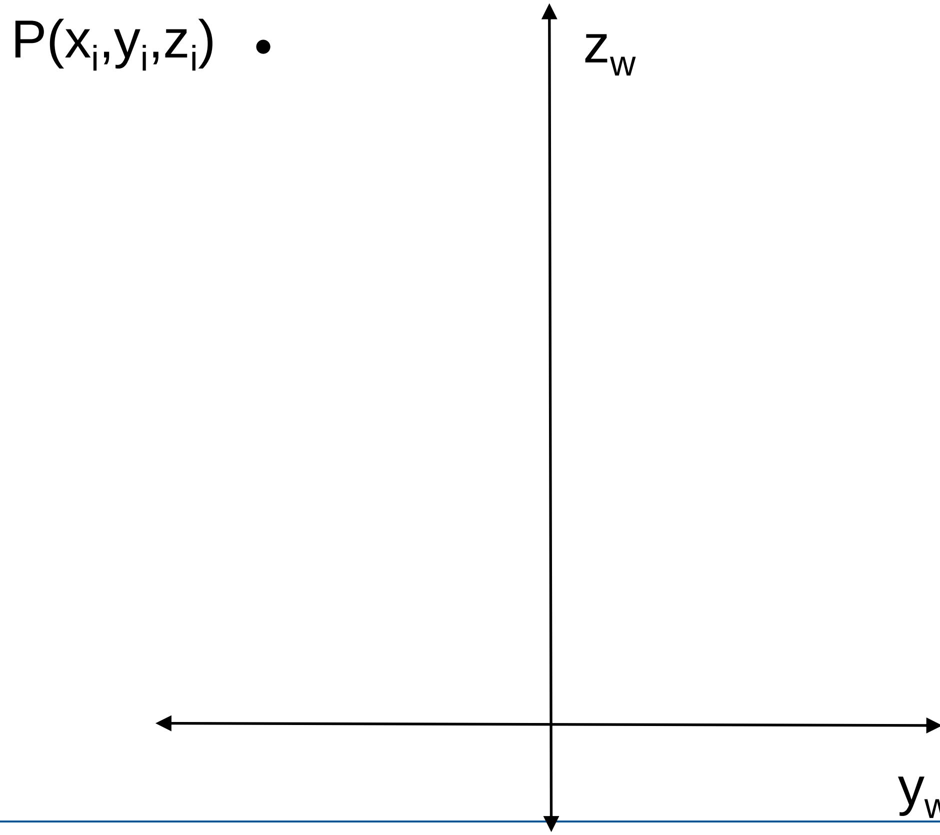
…

3. Two-Stage Camera Calibration

3.3 Stage 2: Compute Focal Length, Distortion and z

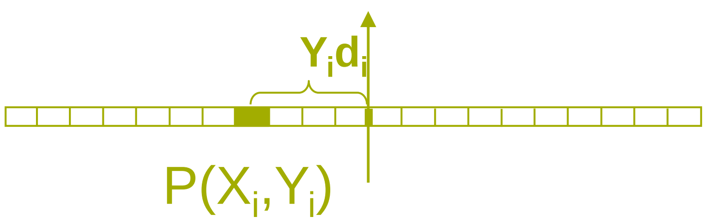
- Interrelationship between f and T_z (ignoring lens distortion)

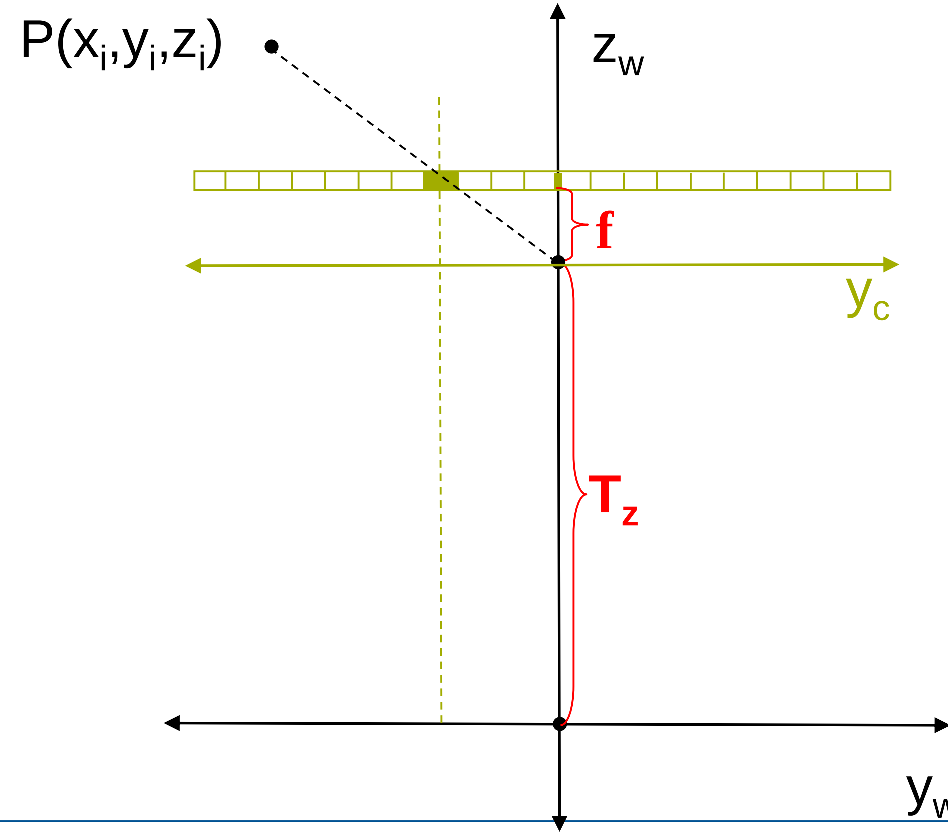




3.3 Stage 2: Compute Focal Length, Distortion and z

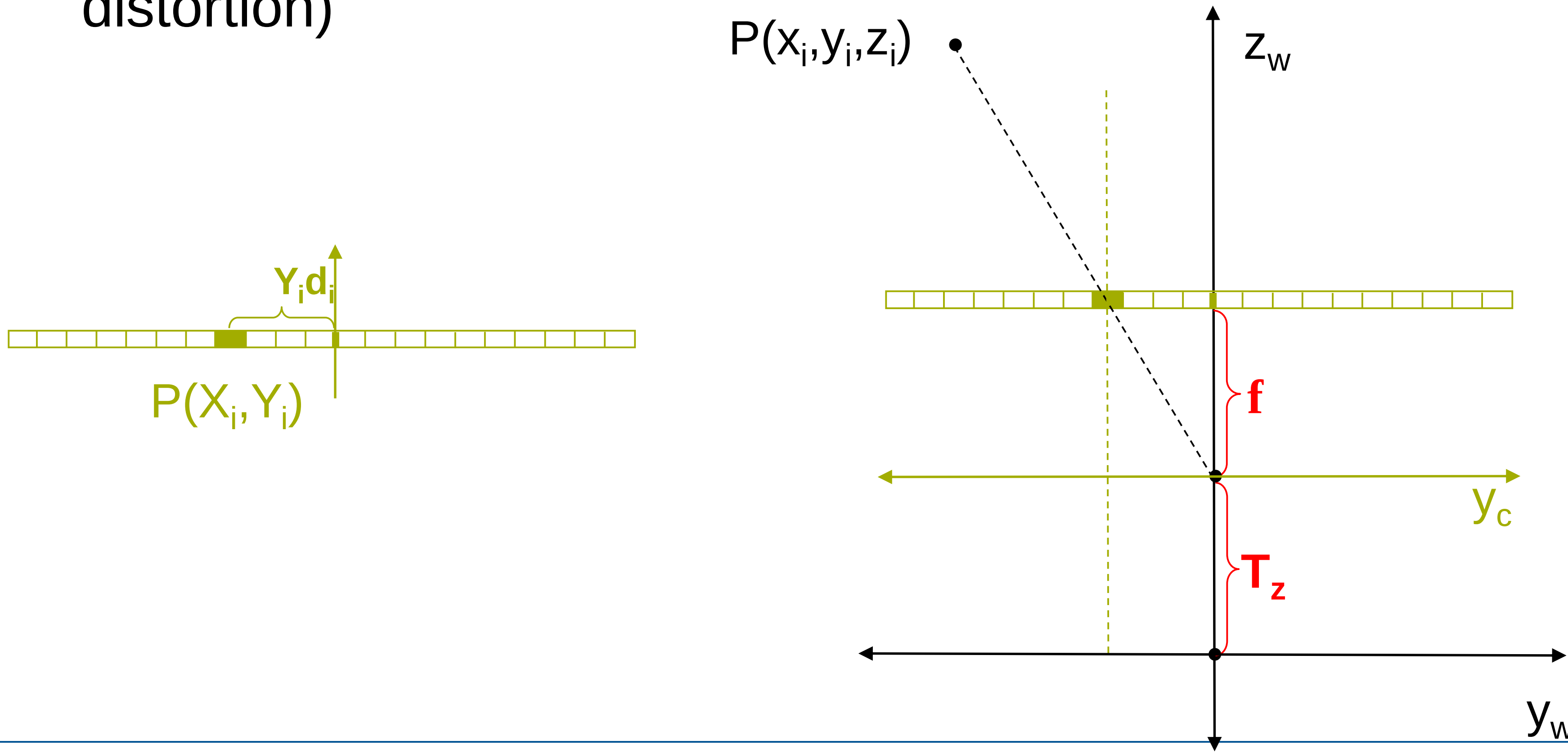
- Interrelationship between f and T_z (ignoring lens distortion)





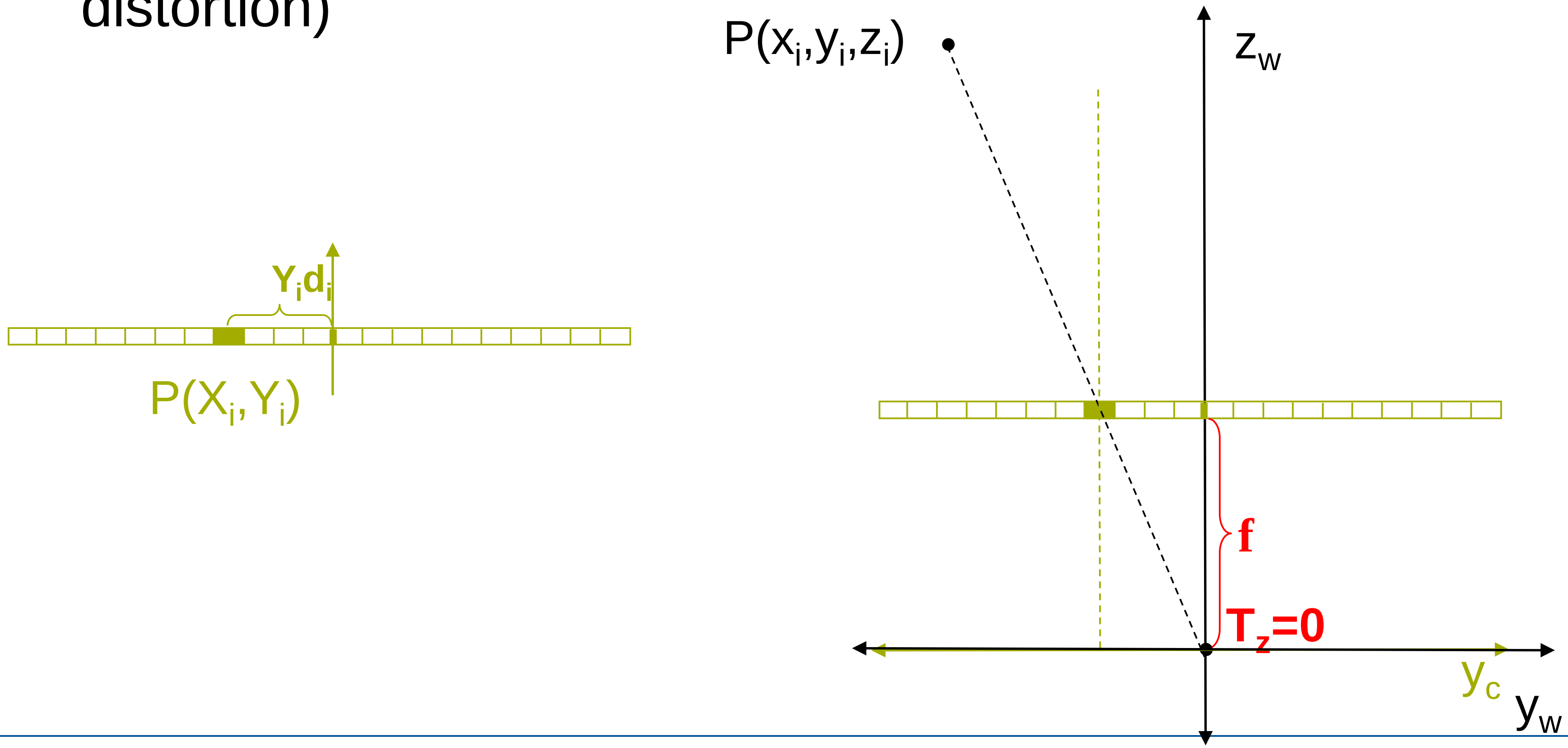
3.3 Stage 2: Compute Focal Length, Distortion and z

- Interrelationship between f and T_z (ignoring lens distortion)



3.3 Stage 2: Compute Focal Length, Distortion and z

- Interrelationship between f and T_z (ignoring lens distortion)



3.3 Stage 2: Compute Focal Length, Distortion and z

Derivation of Formula:

for every marker i:

$$f y_i / z_i = d_y Y_i$$

$$f y_i = z_i (d_y Y_i)$$

$$f y_i - T_z (d_y Y_i) = z_i (d_y Y_i) - T_z (d_y Y_i)$$

$$f y_i - T_z (d_y Y_i) = (z_i - T_z)(d_y Y_i)$$

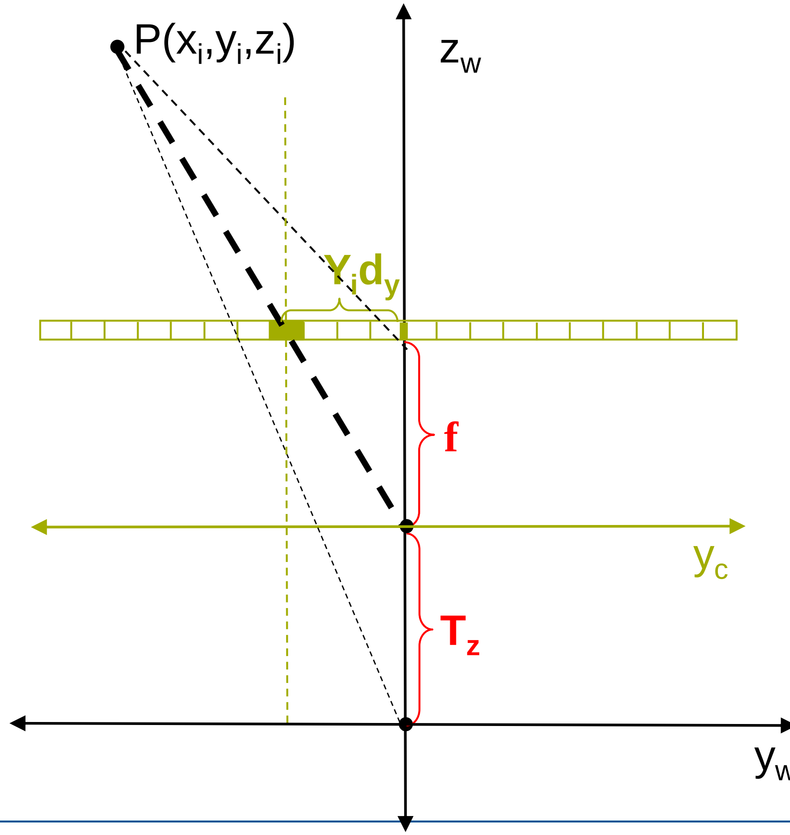
$$f y_i - T_z d_y Y_i = w_i d_y Y_i$$

with:

$$y_i = r_4 x_{wi} + r_5 y_{wi} + r_6 z_{wi} + T_y$$

$$z_i = r_7 x_{wi} + r_8 y_{wi} + r_9 z_{wi} + T_z$$

$$w_i = r_7 x_{wi} + r_8 y_{wi} + r_9 z_{wi} = z_i - T_z$$



3.3 Stage 2: Compute Focal Length, Distortion and z

- Compute an approximation of f and T_z, ignoring lens distortion

For every marker i, formulate

$$y_i f - T_z d_y Y_i = w_i d_y Y_i$$

with:

$$y_i = r_4 x_{wi} + r_5 y_{wi} + r_6 z_{wi} + T_y$$
$$z_i = r_7 x_{wi} + r_8 y_{wi} + r_9 z_{wi} + T_z$$

$$w_i = r_7 x_{wi} + r_8 y_{wi} + r_9 z_{wi} \qquad = z_i - T_z$$

- Compute exact solution for f, T_z, κ_1 and κ_2

3. Two-Stage Camera Calibration

3.4 Summary (Tsai Algorithm)

- World-to-camera transformation: rotation followed by translation
- 6 intrinsic and 6 extrinsic camera parameters
- Parallelism constraint under radial distortion, variable focal length and translation in z
- 2-stage algorithm:
  - Compute 3D orientation, T_x and T_y
  - Compute focal length, distortion coefficients and T_z
  - (other parameters taken from device specifications)
- Separate solutions for coplanar and non-coplanar sets of markers

Agenda

1. Overview
2. Image Formation Steps
3. Two-Stage Camera Calibration
4. How to use Tsai-code with OpenGL

4. How to use Tsai-code with OpenGL

Roger Y. Tsai: A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. IEEE J. Rob. Autom. 3(4): 323-344 (1987).

http://www.vision.caltech.edu/bouguetj/calib_doc/papers/Tsai.pdf

4. How to use Tsai-code with OpenGL

Global Variables: cp, cd, cc

Set of library routines, operating on global variables:

- extern struct camera_parameters
  { Ncx, Nfx, dx, dy, dpx, dpy, Cx, Cy, sx } cp;
- extern struct calibration_data
  { point_count, xw[n], yw[n], zw[n], xf [n], yf[n] } cd;
- extern struct calibration_constants
  { f, kappa1, p1, p2, Tx, Ty, Tz, Rx, Ry, Rz,
  r1, r2, r3, r4, r5, r6, r7, r8, r9 } cc;

4. How to use Tsai-code with OpenGL

Calibration routines

- coplanar_calibration ();
- coplanar_calibration_with_full_optimization ();
- noncoplanar_calibration ();
- noncoplanar_calibration_with_full_optimization ();
- coplanar_extrinsic_parameter_estimation ();
- noncoplanar_extrinsic_parameter_estimation ();

4. How to use Tsai-code with OpenGL

Initialization

Example:

```
cp.Ncx = w;          cp.Nfx = w;
cp.dx = 0.01;        cp.dy = 0.01;
cp.dpx = 1.0;        cp.dpy = 1.0;
cp.Cx = w/2;         cp.Cy = h/2;
cp.sx = 1.0;

for (i=0; i<n; i++) {
    cd.xw[i] = …;   cd.yw[i] = …;   cd.zw[i] = …;
    cd.Xf[i] = …;   cd.Yf[i] = …;
}
```

4. How to use Tsai-code with OpenGL

Interpretation of Results

```
t[0] = cc.Tx;       t[1] = cc.Ty;       t[2] = cc.Tz;

r[0][0] = cc.r1;   r[0][1] = cc.r2;   r[0][2] = cc.r3;
r[1][0] = cc.r4;   r[1][1] = cc.r5;   r[1][2] = cc.r6;
r[2][0] = cc.r7;   r[2][1] = cc.r6;   r[2][2] = cc.r9;

*cx = cc.Cx;
*cy = cc.Cy;
*fx = cc.f * cp.sx;
*fy = cc.f;
*sx = cp.sx;
```

4. How to use Tsai-code with OpenGL

Use of Results with OpenGL

```
dx = (w/2) / fx;                    dy = (h/2) / fy;
alpha_x = 2 * arctan ( dx );        alpha_y = 2 * arctan ( dy );

center_x = 0;                       center_y = 0;
right = dx;                         top = dy;
left = -dx;                         bottom = -dy;
near = 1.0;                         far = 1e6;

glMatrixMode (GL_PROJECTION);
glFrustum (left, right, bottom, top, near, far);
glViewport (0,0,w,h);
```

4. How to use Tsai-code with OpenGL

Use of Results with OpenGL

Translation matrix: Rotation matrix:

double TMat[16]; double RMat[16];

1	0	0	0	r1	-r4	-r7	0	(OpenGL matrices
0	1	0	0	r2	-r5	-r8	0	in column-major order!!!
0	0	1	0	r3	-r6	-r9	0	Rotated coordinate system)
tx	-ty	-tz	1	0	0	0	1	

```
glMatrixMode (GL_MODELVIEW);
glLoadIdentity ();
glMultMatrixd (TMat);
glMultMatrixd (RMat);
```

Thank you!