Bauhaus-Universität Weimar

Photogrammetric Computer Vision

Exercise 4

Winter semester 24/25

(Course materials for internal use only!)

Computer Vision in Engineering – Prof. Dr. Rodehorst
M.Sc. Mariya Kaisheva
mariya.kaisheva@uni-weimar.de

Agenda

Topics

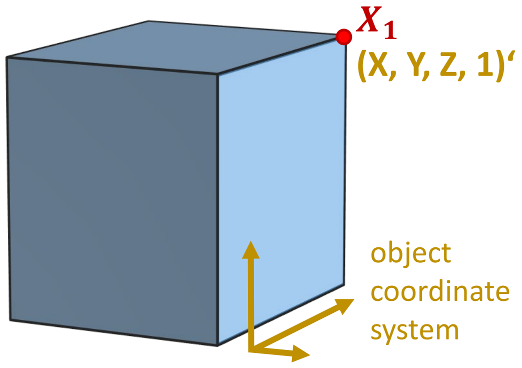
Assignment 1.	Points and lines in the plane, first steps in MATLAB / Octave
Assignment 2.	Projective transformation (Homography)
Assignment 3.	Camera calibration using direct linear transformation (DLT)
Assignment 4.	**Orientation of an image pair**
Assignment 5.	Projective and direct Euclidean reconstruction
Assignment 6.	Stereo image matching
Final Project	- will be announced later -

Agenda

	Start date		Deadline
Assignment 1.	~~21.10.24~~	~~–~~	~~03.11.24~~
Assignment 2.	~~04.11.24~~	~~–~~	~~17.11.24~~
Assignment 3.	~~18.11.24~~	~~–~~	~~01.12.24~~
Assignment 4.	**02.12.24**	**–**	**15.12.24**
Assignment 5.	16.12.24	–	12.01.25
Assignment 6.	13.01.25	–	26.01.25
Final Project.	27.01.25	–	16.03.25

Assignment 3

Assignment 3: *Camera calibration using DLT*



calibration object

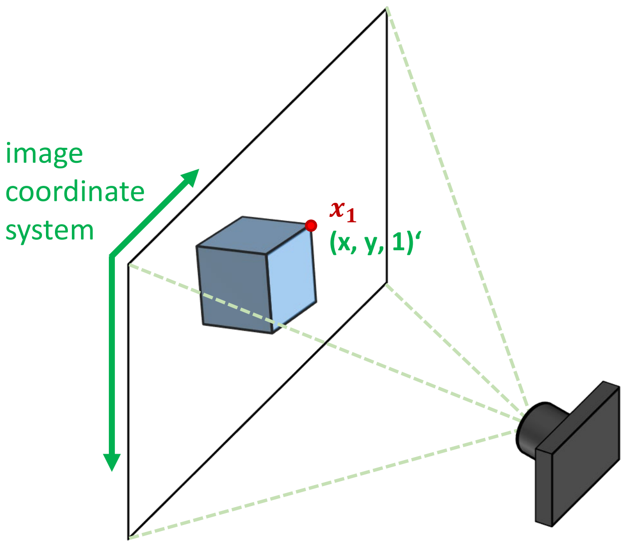


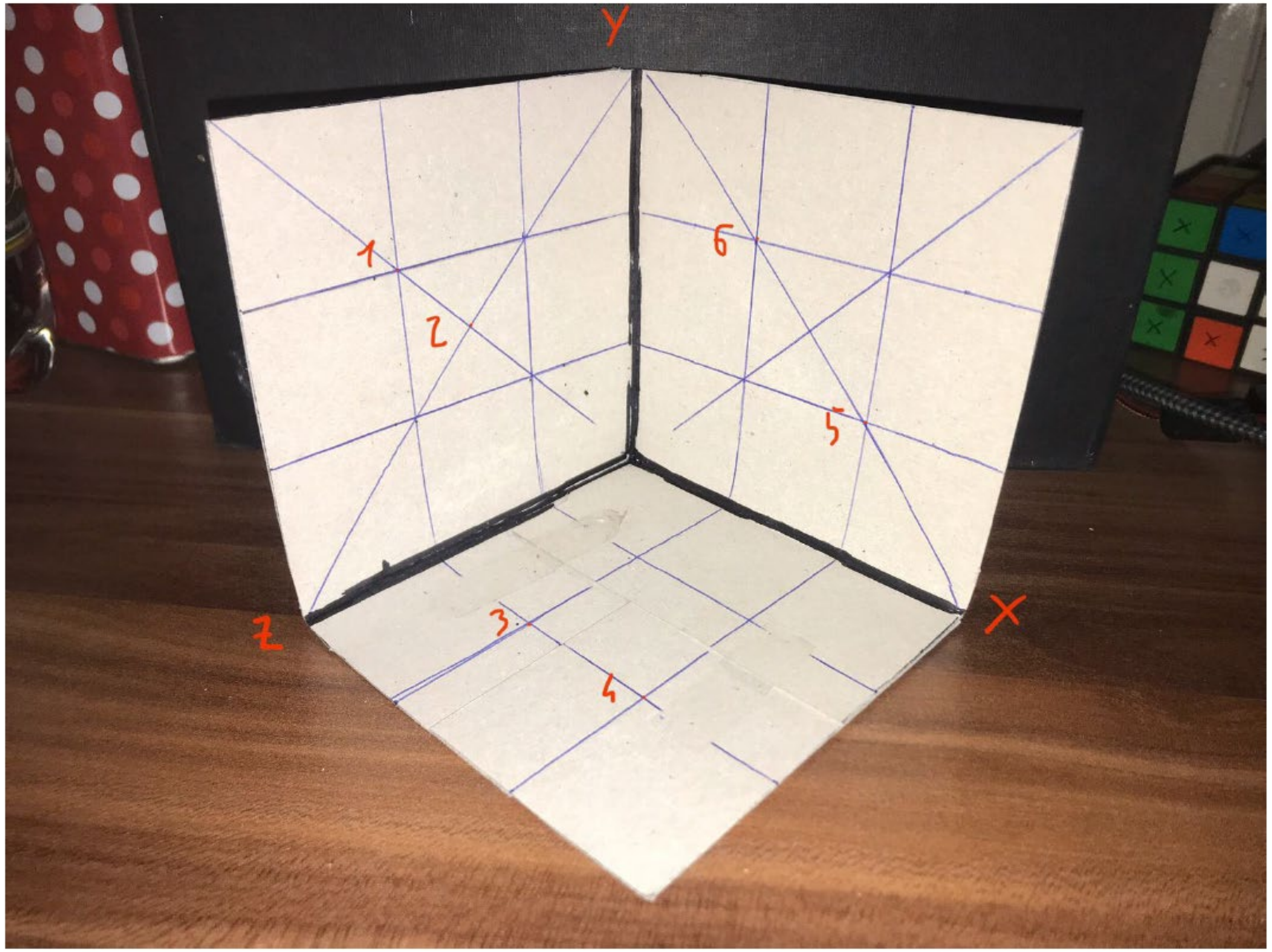
image of the
calibration object

$$x_1 = PX_1$$
$$x_2 = PX_2$$
$$\dots$$
$$x_6 = PX_6$$

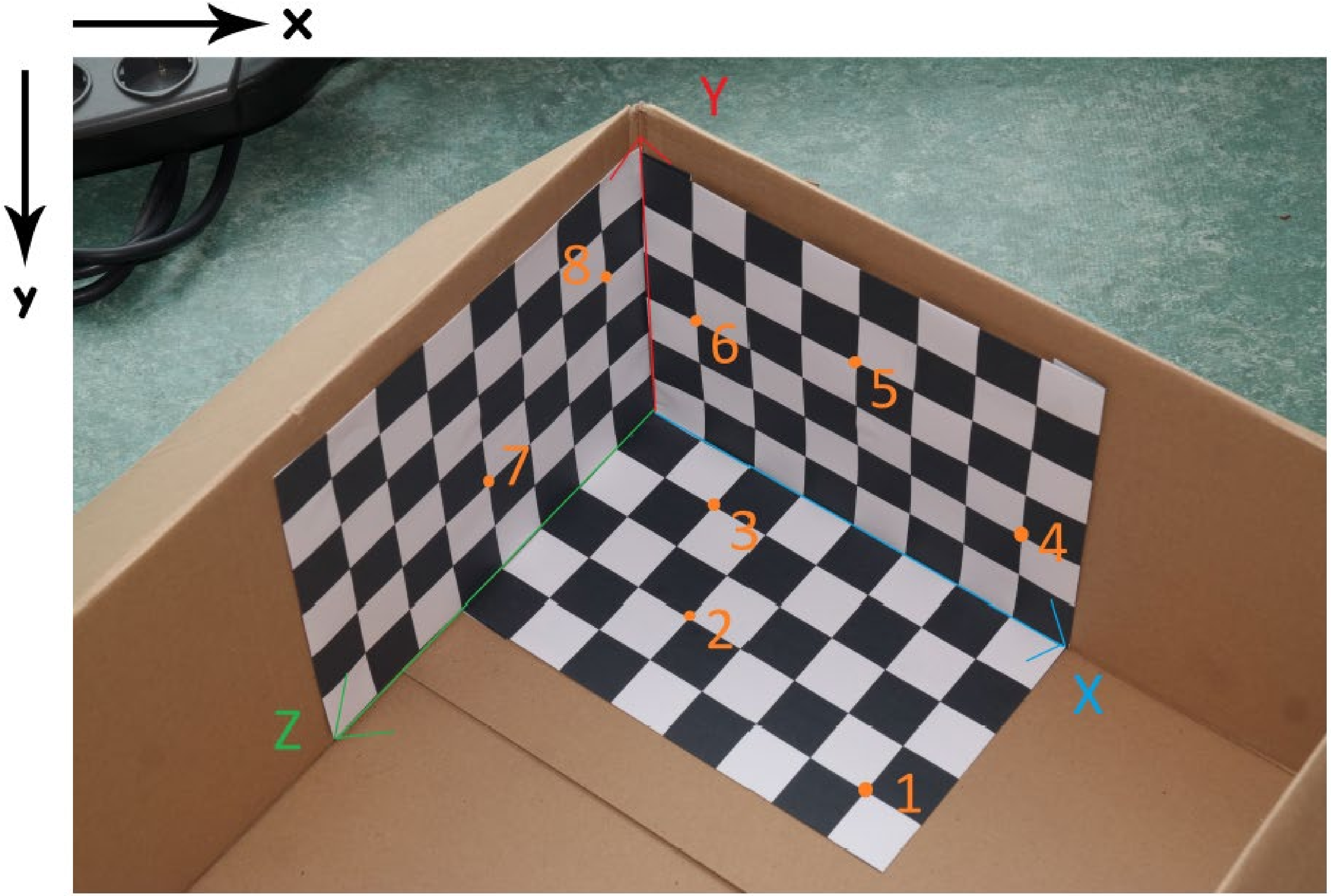
$$x_i = \underbrace{PX_i}_{K\,\&\,R,C}$$

Bauhaus-
Universität
Weimar

Assignment 3 – example calibration objects

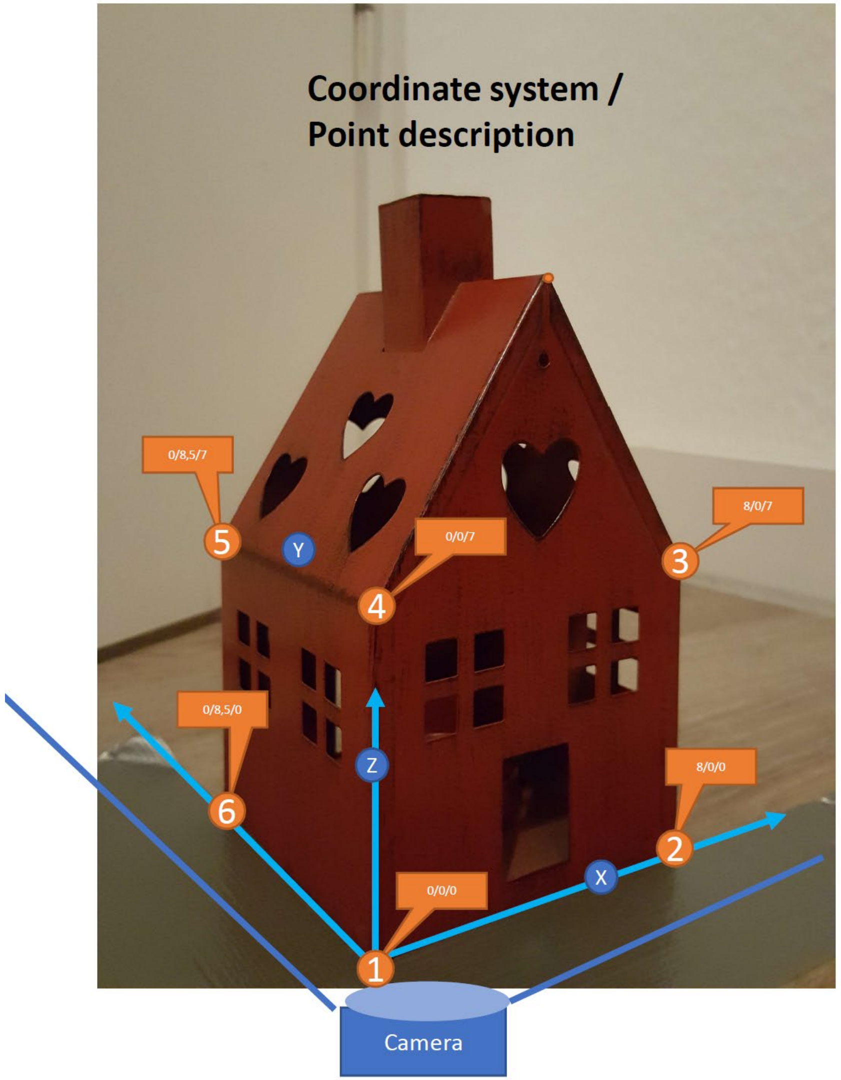


Assignment 3 – example calibration objects



Assignment 3 – example calibration objects

Assignment 3 – example calibration objects



Assignment 3 – sample solution

Bauhaus-
Universität
Weimar

Sample code 1/3

```
% Camera orientation and calibration using direct linear transformation (DLT)

function exercise3
%        =========
X = [  44.7 -103.6   47.4 -152.2 -153.3 -149.4;       % 6 known control points
     -142.4 -146.6 -150.1   59.4  -96.9   52.7;
      258.7  154.4   59.8  245.2  151.3   46.9]; X(4, :) = 1;
x = [  18.5   99.1   13.8  242.1  151.1  243.1;      % 6 measured image points
       46.8  146.5  221.8   52.5  147.1  224.5]; x(3, :) = 1;
P = calibrate(X, x);                                           % Spatial resection
disp_camera(P);                            % Extract and print orientation parameters

function P = calibrate(X, x)
%            ===============
T = condition3(X); N = T * X;                             % Object point conditioning
t = condition2(x); n = t * x;                              % Image point conditioning
A = design_camera(N, n);                                       % Build design matrix
p = solve_dlt(A);                                     % Linear least squares solution
P = inv(t) * reshape(p, 4, 3)' * T                  % Unconditioning and print matrix

function A = design_camera(X, x)     % Design matrix for perspective projection
%            ===================
A = [];
for i = 1 : size(X, 2)
    A = [ A; -x(3, i) * X(:, i)'  0 0 0 0  x(1, i) * X(:, i)' ;
             0 0 0 0  -x(3, i) * X(:, i)'  x(2, i) * X(:, i)' ];
end
```

reshape(p, 4, 3)‘

≠

reshape(p, 3, 4)

Conditioning

```
function T = condition2(x)                    % Conditioning for image points
%             =============
tx = mean(x(1,:));          ty = mean(x(2,:));          % Translation tx, ty
sx = mean(abs(x(1,:) - tx)); sy = mean(abs(x(2,:) - ty));    % Scaling sx, sy
T = [ 1/sx, 0, -tx/sx; 0, 1/sy, -ty/sy; 0, 0, 1 ];
```

```
function T = condition3(X)                    % Conditioning for object points
%             =============
tx = mean(X(1,:));         ty = mean(X(2,:));         tz = mean(X(3,:));
sx = mean(abs(X(1,:)-tx)); sy = mean(abs(X(2,:)-ty)); sz = mean(abs(X(3,:)-tz));
T = [ 1/sx, 0,    0,    -tx/sx;
      0,    1/sy, 0,    -ty/sy;
      0,    0,    1/sz, -tz/sz;
      0,    0,    0,     1 ];
```

Sample code 2/3

```
function disp_camera(P)
%        ===============
M = P(:, 1:3);                                  % M is the left 3x3 submatrix of P
M = M / norm(M(3, :));               % The 3rd row of M defines the scaling factor
if det(M) < 0                              % The determinant of M defines the sign
    M = -M;
end
[K, R] = rqd(M)        % Decomposition of M in calibration and rotation matrix
```

K, R

```
function [R, Q] = rqd(M)     % Decomposition in triangular & orthogonal matrix
%                 ======
[Q, R] = qr(inv(M));                          % RQ decomposition by QR decomposition
R = inv(R); Q = inv(Q);
s = sign(diag(R));                             % Find negative diagonal elements of R
R = R*diag(s);                                    % Invert columns of R and rows of Q
Q = diag(s)*Q;                        % since M = R*Q = R*(-I*-I)*Q = (R*-I)*(-I*Q)
```

Sample code 2/3

```
function disp_camera(P)
%        ===============
M = P(:, 1:3);                                    % M is the left 3x3 submatrix of P
M = M / norm(M(3, :));                  % The 3rd row of M defines the scaling factor
if det(M) < 0                                 % The determinant of M defines the sign
    M = -M;
end
[K, R] = rqd(M)           % Decomposition of M in calibration and rotation matrix
```

K, R

```
function [R, Q] = rqd(M)     % Decomposition in triangular & orthogonal matrix
%                 ======
[Q, R] = qr(inv(M));                           % RQ decomposition by QR decomposition
R = inv(R); Q = inv(Q);
s = sign(diag(R));                             % Find negative diagonal elements of R
R = R*diag(s);                                   % Invert columns of R and rows of Q
Q = diag(s)*Q;                          % since M = R*Q = R*(-I*-I)*Q = (R*-I)*(-I*Q)
```

`diag()`
gets diagonal elements of matrix
or
creates diagonal matrix

Sample code 2/3

`sign(x)`

returns:

1 for x > 0

0 for x = 0

-1 for x < 0

```
function disp_camera(P)
%        ===============
M = P(:, 1:3);                          % M is the left 3x3 submatrix of P
M = M / norm(M(3, :));          % The 3rd row of M defines the scaling factor
if det(M) < 0                        % The determinant of M defines the sign
    M = -M;
end
[K, R] = rqd(M)       % Decomposition of M in calibration and rotation matrix
```

$$\overbrace{}^{K,R}$$

```
function [R, Q] = rqd(M)     % Decomposition in triangular & orthogonal matrix
%                 ======
[Q, R] = qr(inv(M));                    % RQ decomposition by QR decomposition
R = inv(R); Q = inv(Q);
s = sign(diag(R));                      % Find negative diagonal elements of R
R = R*diag(s);                            % Invert columns of R and rows of Q
Q = diag(s)*Q;                % since M = R*Q = R*(-I*-I)*Q = (R*-I)*(-I*Q)
```

Parameter Computation

Projection Center

Solve the property of the projection matrix as linear homogeneous equation system $\mathbf{P} \cdot \mathbf{C} = \mathbf{0}$ using SVD

Orientation Angles

$$\omega = arctan\left(\frac{r_{32}}{r_{33}}\right), \varphi = -\arcsin(r_{31}), \kappa = arctan\left(\frac{r_{21}}{r_{11}}\right)$$

Exterior Orientation

Interior Orientation

Principle distance α_x [in px]

Principle point $(x_0, y_0)^T$

Aspect ratio γ of image axes

shearing angle θ of image axes

$$\mathbf{K} = \begin{bmatrix} ck_x & -ck_x \cot(\theta) & x_0 \\ 0 & ck_y / \sin(\theta) & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

physical model

$$\mathbf{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

algebraic model

Parameter Computation

Projection Center

Solve the property of the projection matrix as linear homogeneous equation system $\mathbf{P} \cdot \mathbf{C} = \mathbf{0}$ using SVD

Orientation Angles

$$\omega = arctan\left(\frac{r_{32}}{r_{33}}\right), \varphi = -\arcsin(r_{31}), \kappa = arctan\left(\frac{r_{21}}{r_{11}}\right)$$

Exterior Orientation

Interior Orientation

Principle distance α_x [in px]

Principle point $(x_0, y_0)^{\mathsf{T}}$

Aspect ratio γ of image axes

shearing angle θ of image axes

$$\mathbf{K} = \begin{bmatrix} ck_x & -ck_x \cot(\theta) & x_0 \\ 0 & ck_y / \sin(\theta) & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

physical model

$$\mathbf{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

algebraic model

Parameter Computation

Projection Center

Solve the property of the projection matrix as linear homogeneous equation system $\mathbf{P} \cdot \mathbf{C} = \mathbf{0}$ using SVD

Orientation Angles

$$\omega = arctan\left(\frac{r_{32}}{r_{33}}\right), \varphi = -\arcsin(r_{31}), \kappa = arctan\left(\frac{r_{21}}{r_{11}}\right)$$

Exterior Orientation

Interior Orientation

Principle distance α_x [in px]

Principle point $(x_0, y_0)^{\mathrm{T}}$

Aspect ratio γ of image axes

shearing angle θ of image axes

$$\mathbf{K} = \begin{bmatrix} ck_x & -ck_x \cot(\theta) & x_0 \\ 0 & ck_y / \sin(\theta) & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

physical model

$$\mathbf{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

algebraic model

Parameter Computation

Projection Center

Solve the property of the projection matrix as linear homogeneous equation system $\mathbf{P} \cdot \mathbf{C} = \mathbf{0}$ using SVD

Orientation Angles

$$\omega = arctan\left(\frac{r_{32}}{r_{33}}\right), \varphi = -\arcsin(r_{31}), \kappa = arctan\left(\frac{r_{21}}{r_{11}}\right)$$

Exterior Orientation

Interior Orientation

Principle distance α_x [in px]

Principle point $(x_0, y_0)^\mathsf{T}$

Aspect ratio γ of image axes

shearing angle θ of image axes

$$\mathbf{K} = \begin{bmatrix} ck_x & -ck_x \cot(\theta) & x_0 \\ 0 & ck_y / \sin(\theta) & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

physical model

$$\mathbf{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

algebraic model

Parameter Computation

Projection Center

Solve the property of the projection matrix as linear homogeneous equation system $\mathbf{P} \cdot \mathbf{C} = \mathbf{0}$ using SVD

Orientation Angles

$$\omega = arctan\left(\frac{r_{32}}{r_{33}}\right), \varphi = -\arcsin(r_{31}), \kappa = arctan\left(\frac{r_{21}}{r_{11}}\right)$$

Exterior Orientation

Interior Orientation

Principle distance α_x [in px]

Principle point $(x_0, y_0)^\mathsf{T}$

Aspect ratio γ of image axes

shearing angle θ of image axes

$$\mathbf{K} = \begin{bmatrix} ck_x & -ck_x \cot(\theta) & x_0 \\ 0 & ck_y / \sin(\theta) & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

physical model

$$\mathbf{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

algebraic model

Sample code 3/3

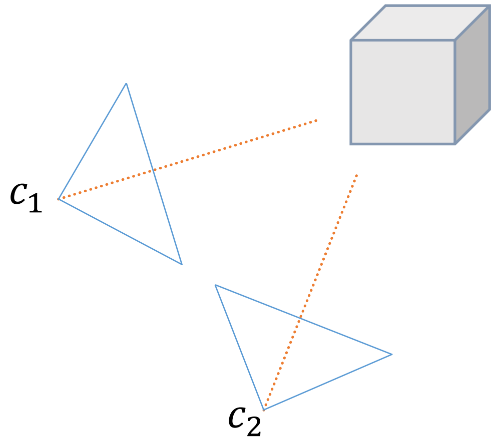
```
function disp_camera(P)
%         ===============
M = P(:, 1:3);                                      % M is the left 3x3 submatrix of P
M = M / norm(M(3, :));                    % The 3rd row of M defines the scaling factor
if det(M) < 0                                   % The determinant of M defines the sign
    M = -M;
end
[K, R] = rqd(M)           % Decomposition of M in calibration and rotation matrix

C = solve_dlt(P); C(1:3)/C(end)  % Estimation of the projection center P C = 0

omega =  atan2(R(3,2), R(3,3)) * 180/pi                   % Print three spatial angles
phi   = -asin(R(3,1)) * 180/pi
kappa =  atan2(R(2,1), R(1,1)) * 180/pi

s     =  acot(-K(1,2) / K(1,1)) * 180/pi                 % Shearing of the image axes
a     =  K(2,2) / K(1,1)                                     % and the aspect ratio
```

Assignment 4: *Orientation of an image pair*

Assignment 4: *Orientation of an image pair*

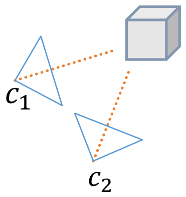
1) Image acquisition

- take pictures of the same **spatially structured** object from **two different views**
- use **convergent image arrangement**

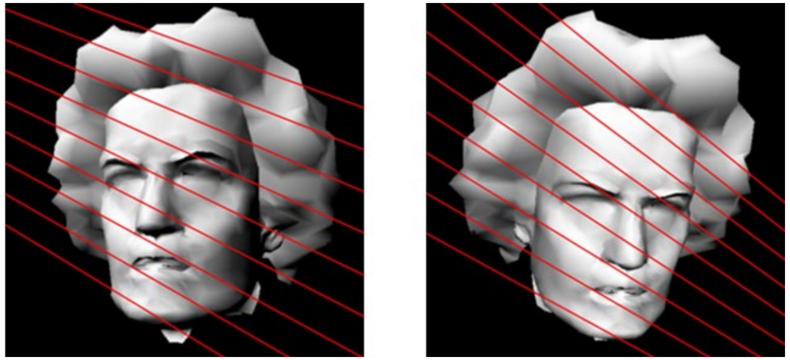


Assignment 4: *Orientation of an image pair*

1) Image acquisition
    - take pictures of the same **spatially structured** object from **two different views**
    - use **convergent image arrangement**



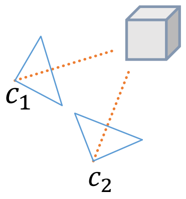
2) Image pair orientation
    - measure **at least 8** point pairs
    - compute the **fundamental matrix F**
    - visualize used points
    - draw **epipolar lines** (*hline.m*)



Assignment 4: *Orientation of an image pair*

1) Image acquisition

   - take pictures of the same **spatially structured** object from **two different views**
   - use **convergent image arrangement**



2) Image pair orientation

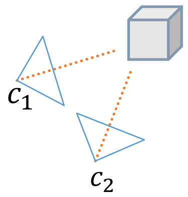
   - measure **at least 8** point pairs
   - compute the **fundamental matrix F**
   - visualize used points
   - draw **epipolar lines** (*hline.m*)

```
% Example call of the HLINE-function
% ----------------------------------
% function test
% l = [-1 1 50]';                     % Line equation in implicit form
% f = imread('image.jpg');            % Read image f
% figure, imshow(f), hold on          % Show image in new window
% hline(l);                           % Draw red line l in image f
```

Assignment 4: *Orientation of an image pair*

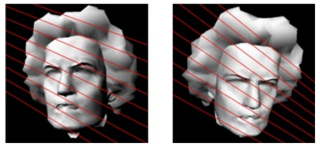
1) Image acquisition

- take pictures of the same **spatially structured** object from **two different views**
- use **convergent image arrangement**



2) Image pair orientation

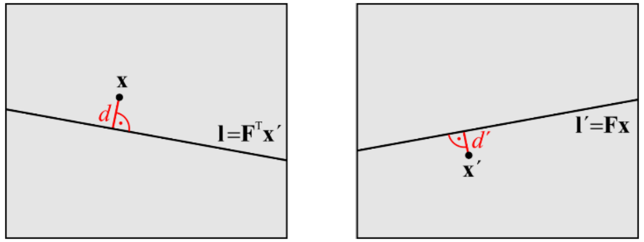
- measure **at least 8** point pairs
- compute the **fundamental matrix F**
- visualize used points
- draw **epipolar lines** (*hline.m*)



$$\sum_i d\left(\mathbf{x}'_i, \mathbf{F}\mathbf{x}_i\right)^2 + d\left(\mathbf{x}_i, \mathbf{F}^{\mathsf{T}}\mathbf{x}'_i\right)^2$$

3) Evaluation

- comment line characteristics
- calculate the **geometric image error**



Bauhaus-Universität Weimar

Assignment 4: *Orientation of an image pair*

Bauhaus-Universität Weimar