Photogrammetric Computer Vision

Exercise 5

Winter semester 24/25

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

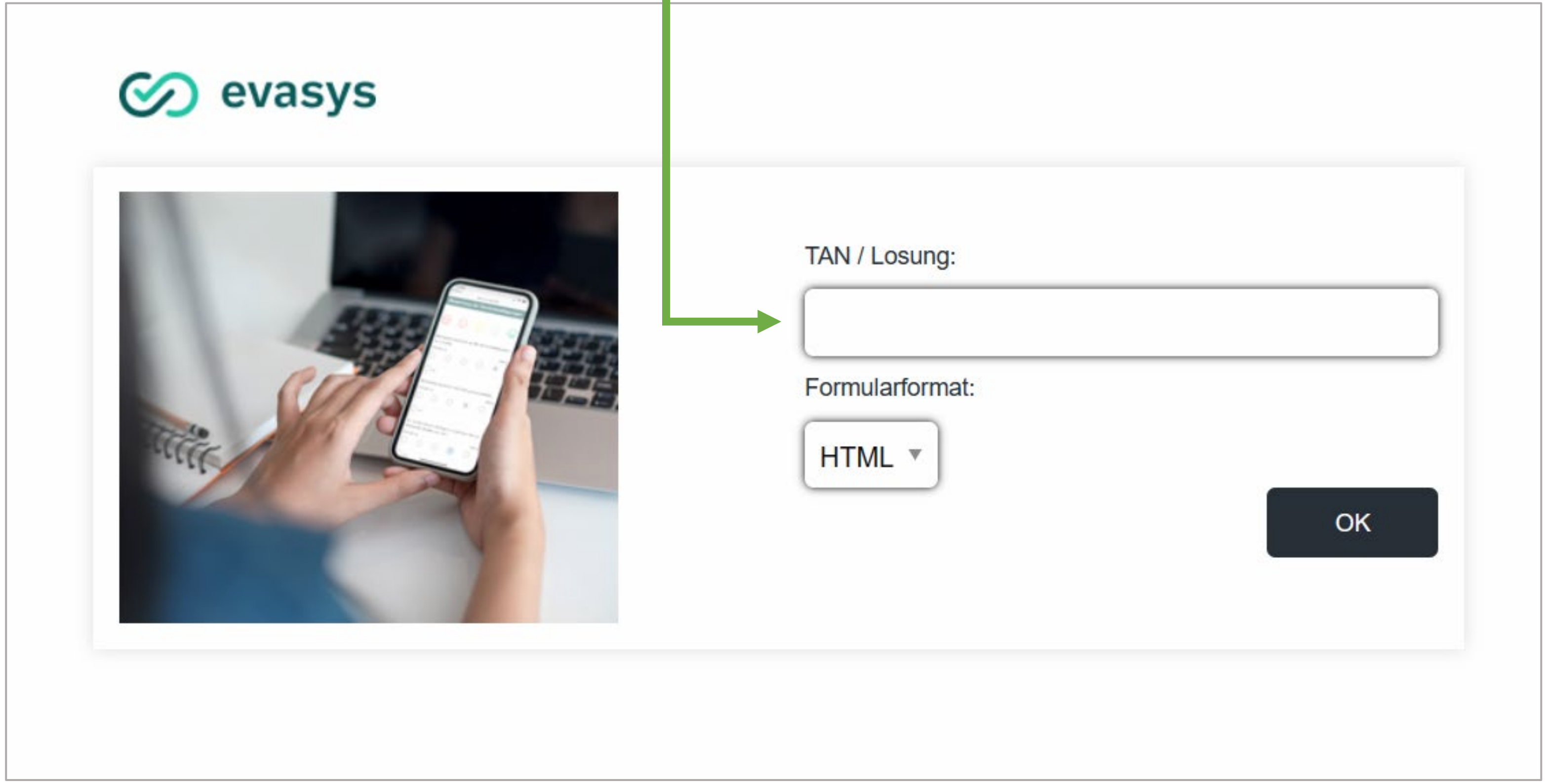
	Start date		Deadline
Assignment 1.	~~21.10.24~~	~~–~~	~~03.11.24~~
Assignment 2.	~~04.11.24~~	~~–~~	~~17.11.24~~
Assignment 3.	~~18.11.24~~	~~–~~	~~01.12.24~~
Assignment 4.	~~02.12.24~~	~~–~~	~~15.12.24~~
Assignment 5.	**16.12.24**	**–**	**12.01.25**
Assignment 6.	13.01.25	–	26.01.25
Final Project.	27.01.25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

Bauhaus-Universität Weimar

Online Evaluation

URL: https://evasys.uni-weimar.de/evasys/online/

Code: **M4H47**



Bauhaus-
Universität
Weimar

Online Evaluation

evasys

Bauhaus-Universität Weimar | WiSe 24/25

Fakultät Medien

Prof. Dr.-Ing. habil. Volker Rodehorst |

Photogrammetric Computer Vision

1 2 3 4 5 7 8 9

Introduction

Dear Bauhaus-Universität Weimar student,

Also this semester we would like to ask you for your feedback on the courses. This will help us to continuously improve the quality of our teaching. Therefore we kindly ask you to participate.

Would you give us feedback? ○ Yes ○ No

Your answers will of course be treated anonymously. Surveys with fewer than 5 answers are not forwarded to the teachers.

Assignment 4: *Orientation of an image pair*

1) Image acquisition

2) Image pair orientation
   - Selection of homologeous points
   - Computation of the Fundamental Matrix **F** using the **normalized 8-point-algorithm**
   - Plotting of the epipolar lines
3) Evaluation
   - Computation of the geometric image error

Assignment 4 – sample solution

Bauhaus-
Universität
Weimar

Sample code 1/3

```
function exercise4
%        =========
f = imread('img1.jpg');                   % Read image pair
g = imread('img2.jpg');
F = relative_orientation(f, g);           % Compute relative orientation


function F = relative_orientation(f, g)
%            ==========================
figure(1); imshow(f); x1 = get_points     % Display images and
figure(2); imshow(g); x2 = get_points     % measure >=8 image points
F = linear_fund(x1, x2)                   % Estimate fundamental matrix

...
```

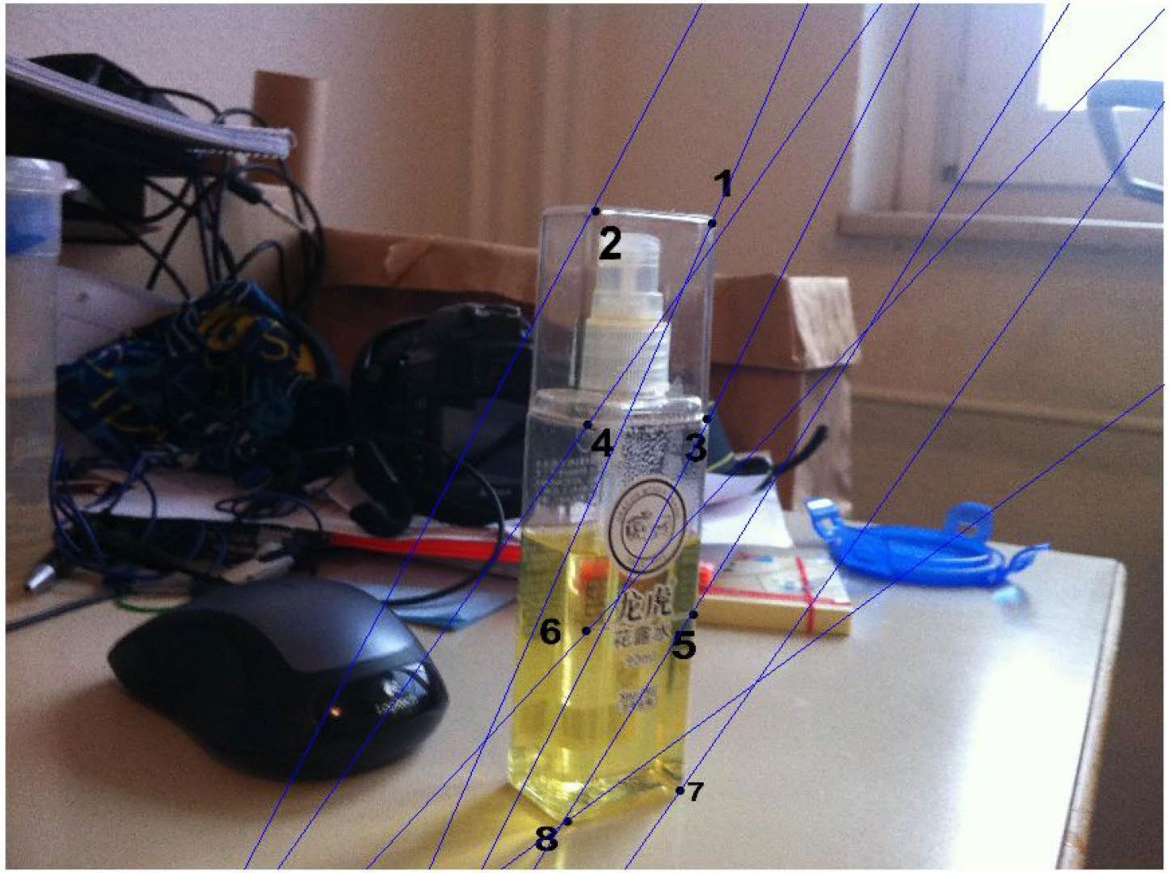
Sample code 2/3

```
function F = linear_fund(x1, x2)          % Normalized 8-point algorithm
%            =====================
T1 = condition2(x1); n1 = T1 * x1;        % Image point conditioning
T2 = condition2(x2); n2 = T2 * x2;
A = design_fund(n1, n2);                  % Build design matrix
f = solve_dlt(A);                         % Linear least-squares-solution
F = reshape(f, 3, 3)';                    % Solution vector in matrix form
F = T2' * force_rank2(F) * T1;            % Force singularity, reverse conditioning

function A = design_fund(x1, x2)     % Design matrix
%            =====================
A = [];
for i = 1 : size(x1, 2)
    A = [ A; x2(1, i)*x1(:, i)' x2(2, i)*x1(:, i)' x2(3, i)*x1(:, i)' ];
end

function F = force_rank2(F)     % Force singularity constraint det(F)=0
%            ===============
[U, D, V] = svd(F);             % Singular value decomposition
D(3, 3) = 0;                    % Smallest singular value must be 0
F = U * D * V';                 % Recompose matrices
```

Sample code 3/3

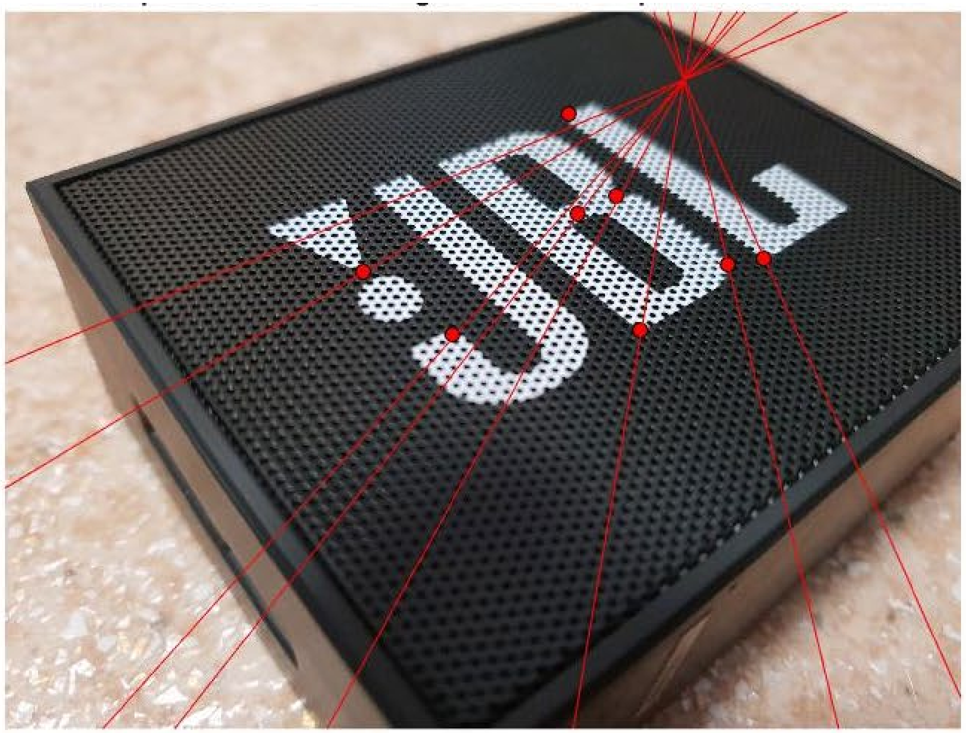
```
function F = relative_orientation(f, g)
%            ==========================
figure(1); imshow(f); x1 = get_points      % Display images and
figure(2); imshow(g); x2 = get_points      % measure >=8 image points
F = linear_fund(x1, x2)                    % Estimate fundamental matrix
figure(1); draw_epipol(x1, F' * x2);       % Draw points and
figure(2); draw_epipol(x2, F * x1);        % epipolar lines
sampson_error(F, x1, x2);                  % Print error estimate

function draw_epipol(x, l)
%        =================
hold on                                    % Draw on existing image
for i = 1 : size(x, 2)
    hline(l(:, i));                        % Draw homogeneous line
    plot(x(1, i), x(2, i), 'ko', 'MarkerFaceColor', 'r');   % Draw point
end

function err = sampson_error(F, x1, x2)          % First order geometric error
%              ========================
l2 = F * x1;                                            % Epipolar lines
l1 = F' * x2;
num = sum(x2 .* l2).^2;                                 % Fraction numerator
den = l2(1,:).^2 + l2(2,:).^2 + l1(1,:).^2 + l1(2,:).^2; % Denominator
err = sum(num ./ den)                     % Final epipolar distance
```

Assignment 4 – example results



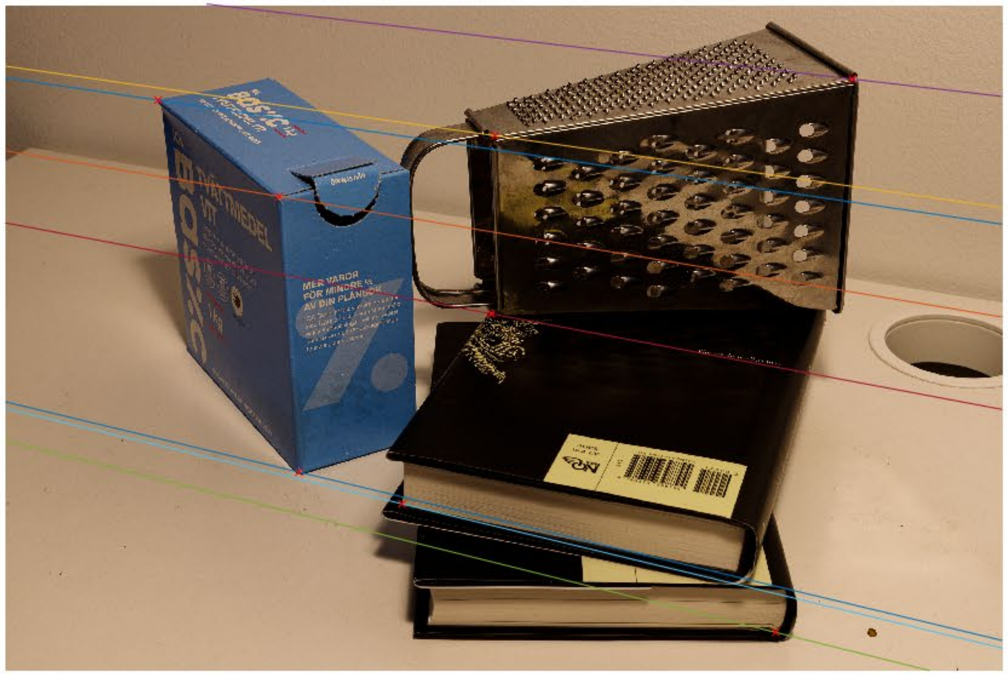
Degenerated Configuration + No Singularity Constraint on F

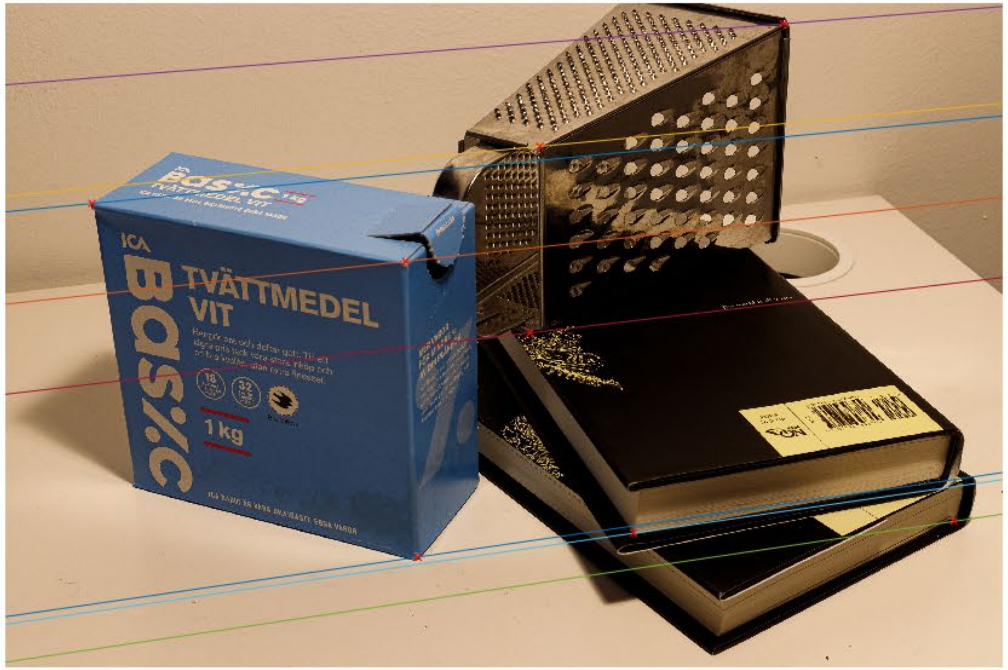
Assignment 4 – example results



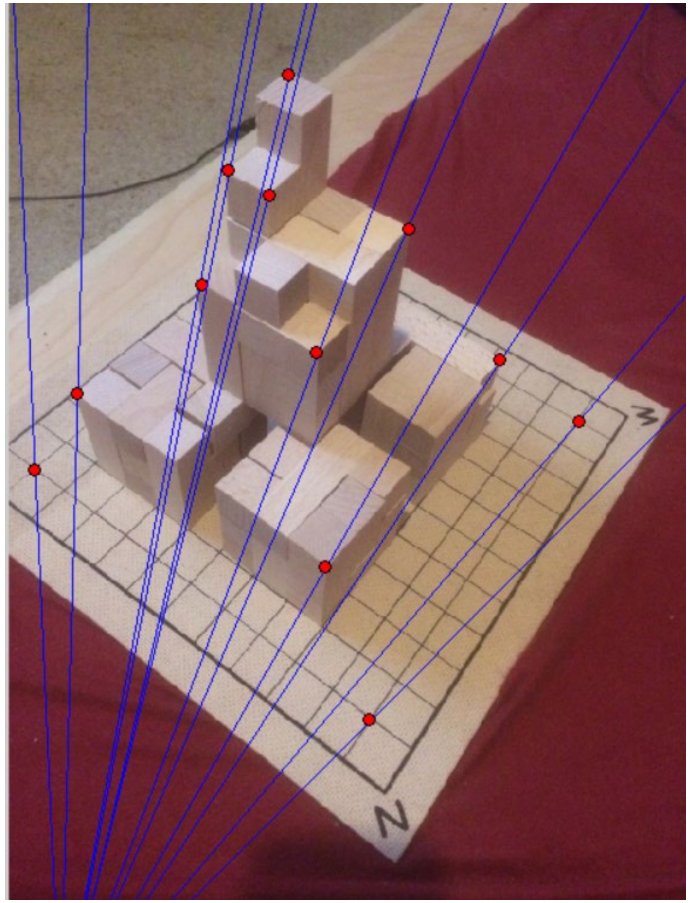
Degenerated Configuration

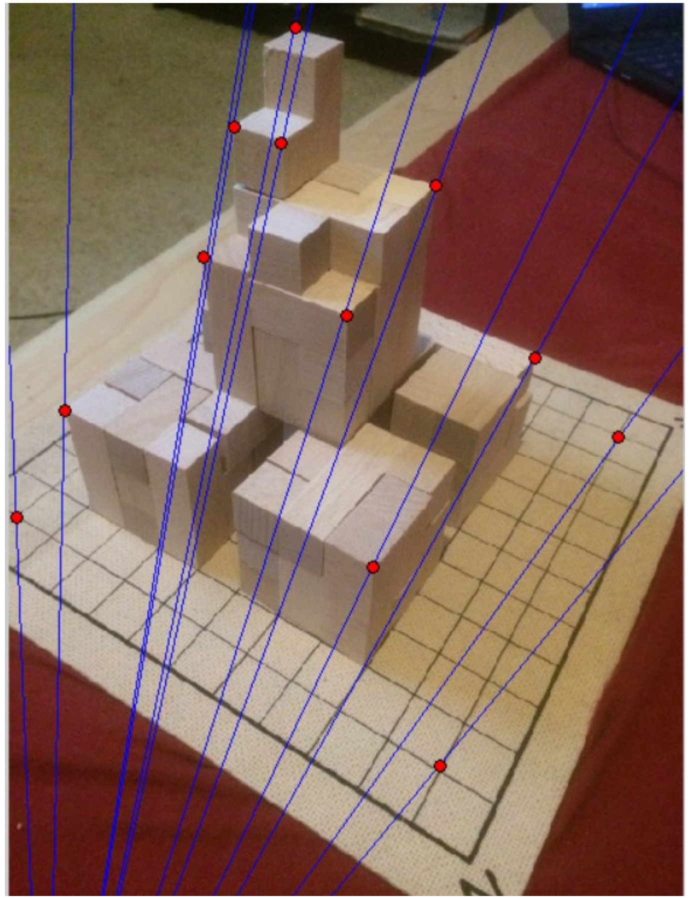
Assignment 4 – example results





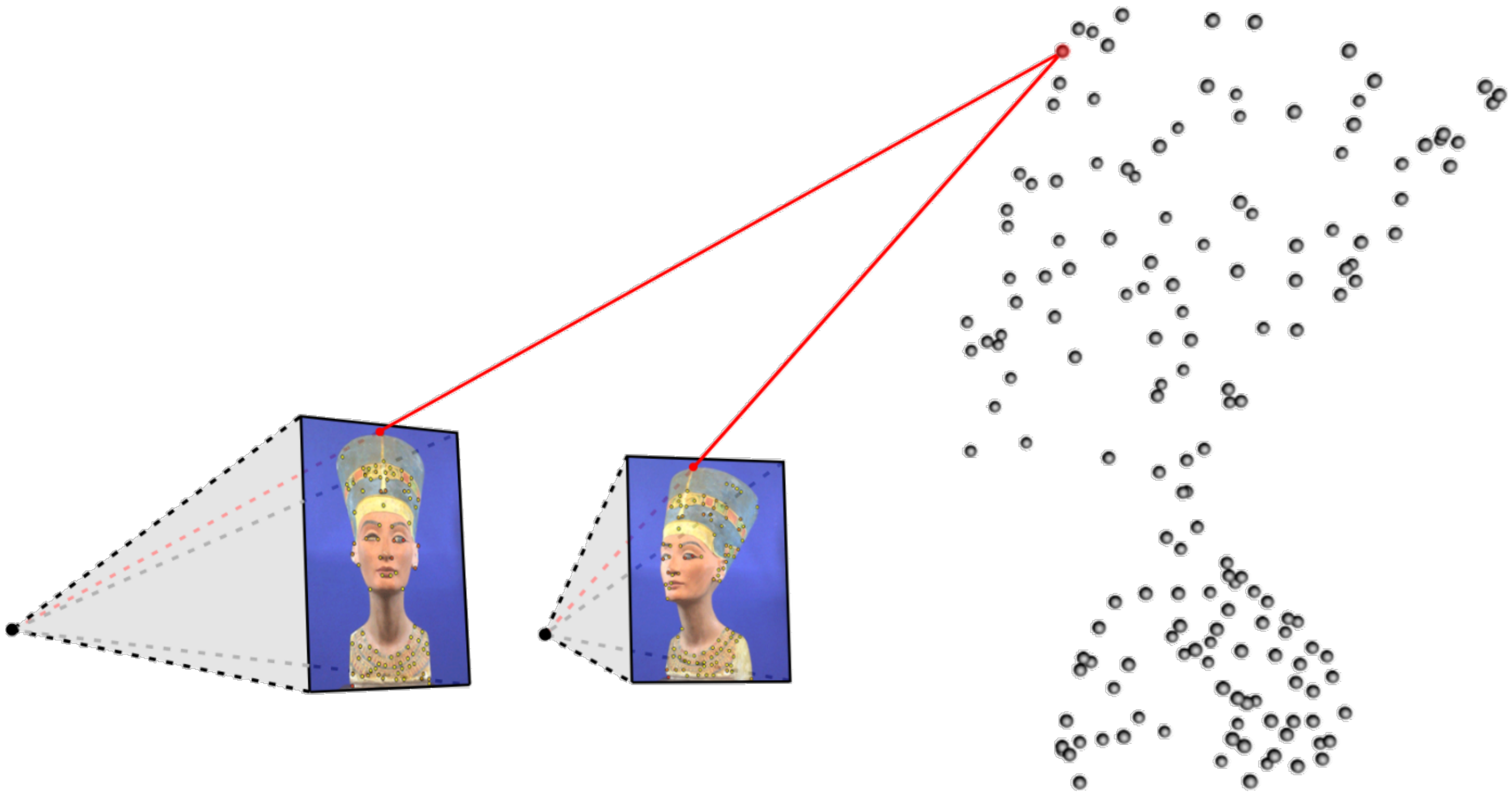
Assignment 4 – example results





Geometric error = 0.45, 12 points

Assignment 5: *Projective and direct Euclidean reconstruction*



Assignment 5: *Projective and direct Euclidean reconstruction*

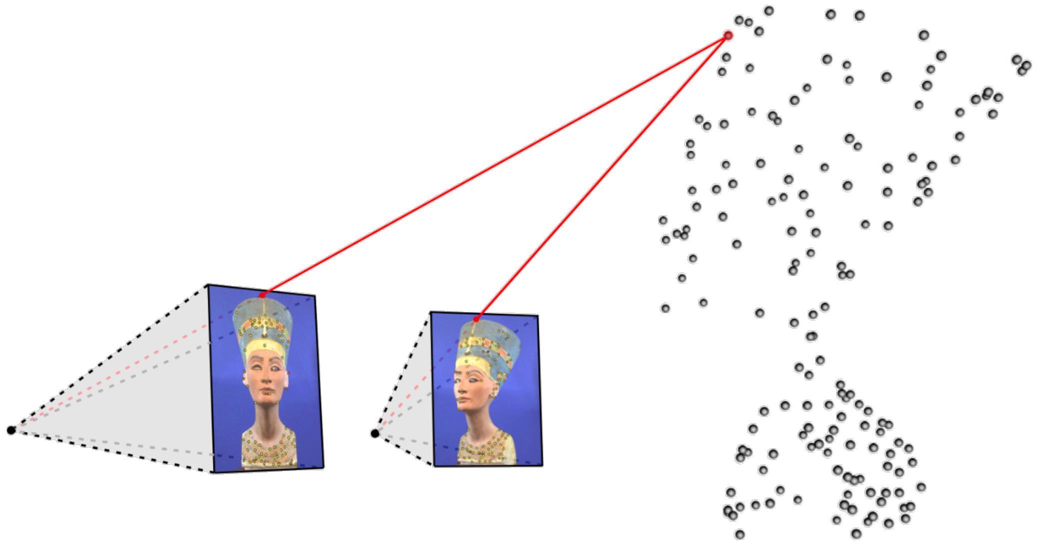
Part 1: **Projective** reconstruction

- Given: homologous image coordinate pairs (*bh.dat*)
- Wanted: triangulation of projective object points

Part 2: **Direct Euclidean** reconstruction

- Given: 5 control points (*pp.dat*)

  + triangulated* object points (Part 1, c)
- Wanted: triangulation of Eucledian object points



* up to a projective transformation

Assignment 5 Part 1: *Projective reconstruction*

1. **Projective reconstruction:**

   Since the manual matching of image points is quite laborious and boring, a text file `bh.dat` with many homologous image points is made available for the image pair showing the bust of BEETHOVEN.

   a) Read the homologous image coordinates $\mathbf{x}_1 \leftrightarrow \mathbf{x}_2$ in the format (x_1, y_1, x_2, y_2), e.g. with

   ```
   fh = fopen('bh.dat', 'r');
   A = fscanf(fh, '%f%f%f%f', [4 inf]);
   fclose(fh);
   x1 = A(1:2, :); x2 = A(3:4, :);
   ```

   and use your function from exercise 4 in order to determine the relative orientation of the images with the *fundamental matrix* $\mathbf{F}$.

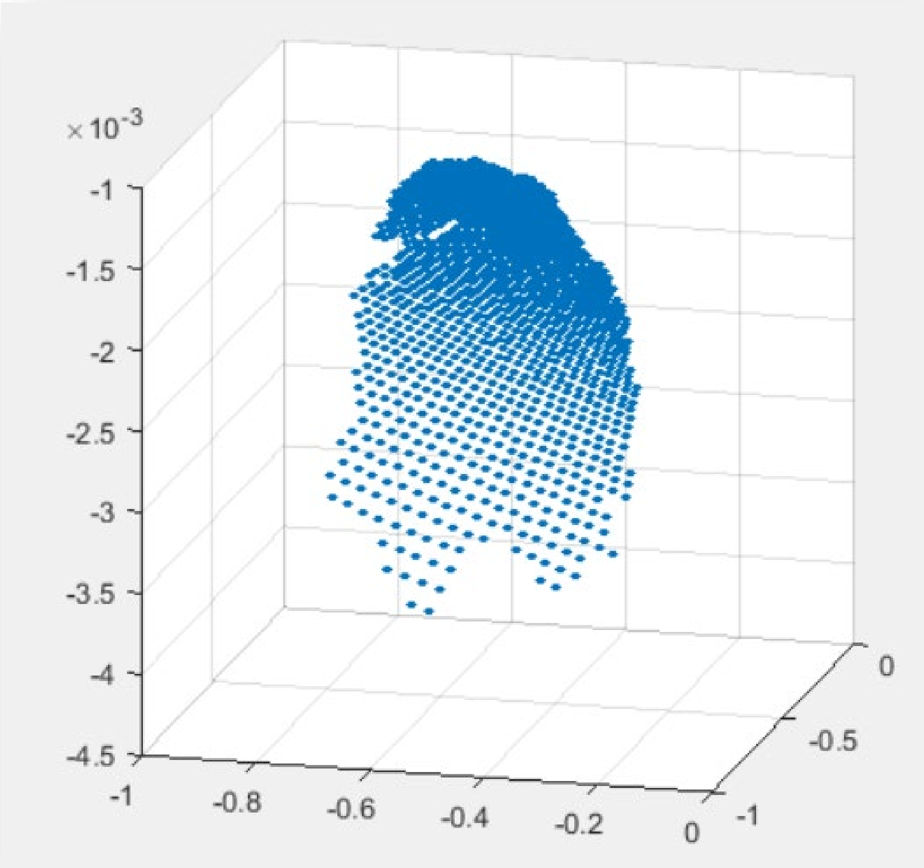
   b) Implement a new function, which defines two corresponding *projection matrices* $\mathbf{P}_N$ and $\mathbf{P}'$ by means of $\mathbf{F}$.

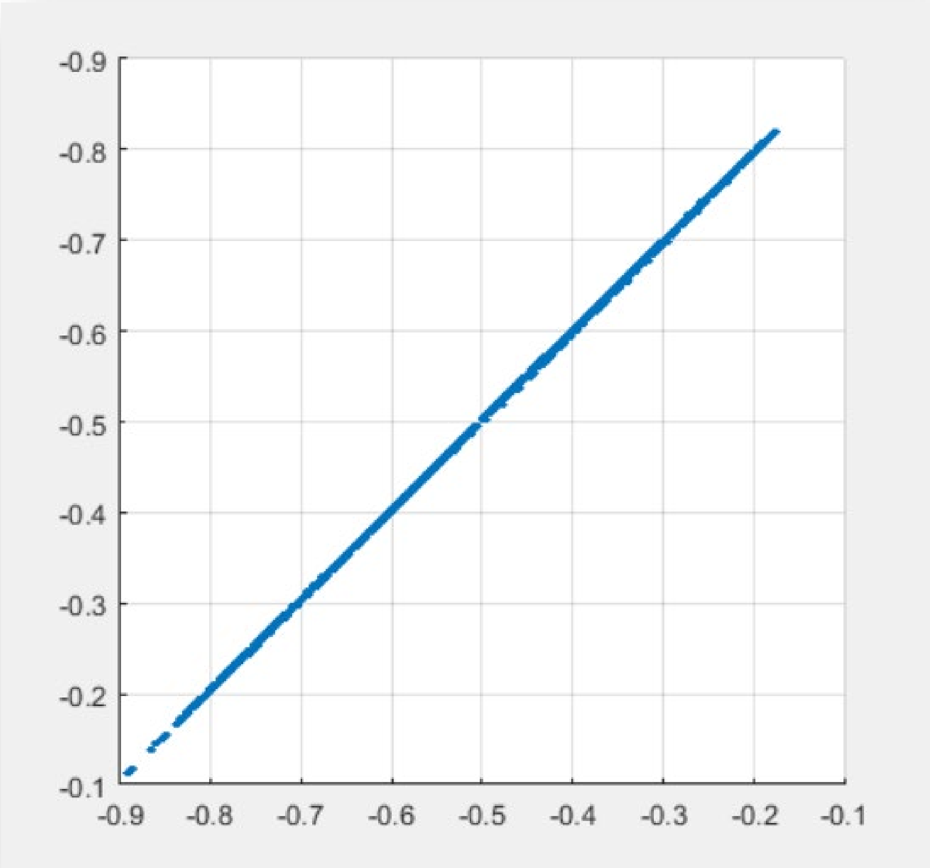
   c) Realize a function for the *linear triangulation* of projective object points $\mathbf{X}_{P1}$ and try to visualize the computed spatial object coordinates, e.g. using

   ```
   figure; scatter3(X(1,:), X(2,:), X(3,:), 10, 'filled');
   axis square; view(32, 75);
   ```

Assignment 5 Part 1: *Projective reconstruction*

Example Results after **Part 1**





Assignment 5 Part 2: *Direct Euclidean reconstruction*

2. Direct Euclidean reconstruction:

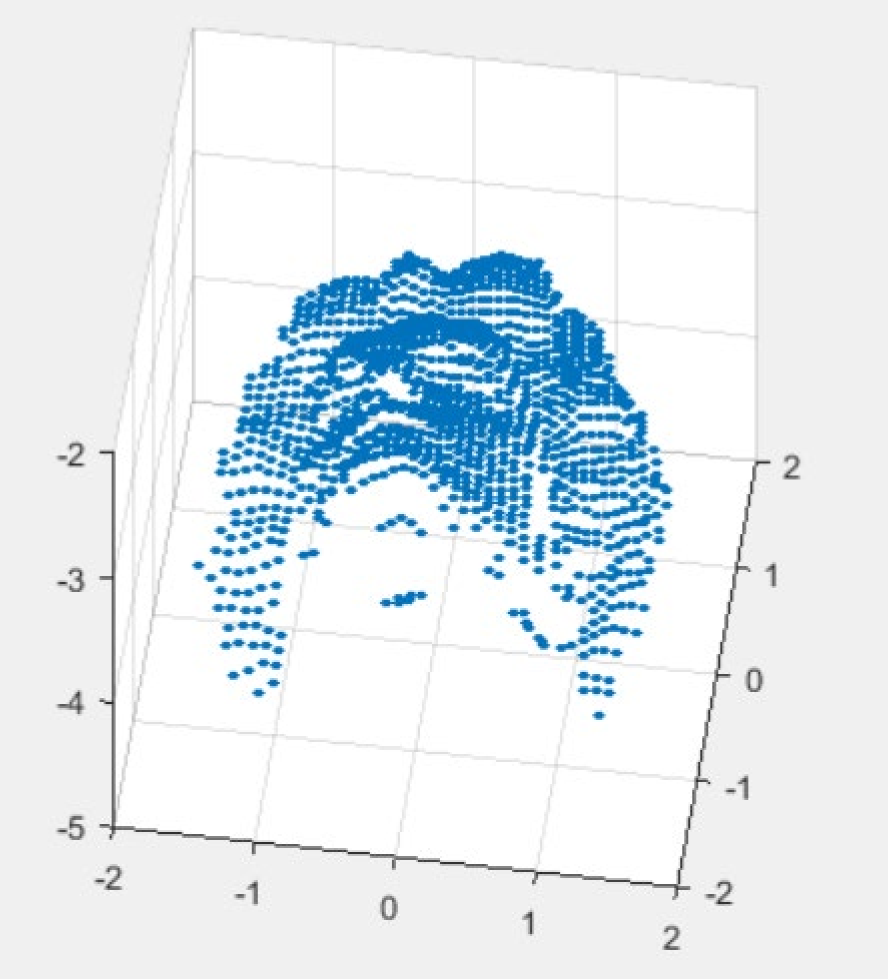
a) Read the *control point* information from the provided file `pp.dat` in the format (x_1, y_1, x_2, y_2, X_E, Y_E, Z_E) and triangulate projective object points $\mathbf{X}_{P2}$ from the five homologous image points $\mathbf{x}_1 \leftrightarrow \mathbf{x}_2$ using the already computed projection matrices $\mathbf{P}_N$ and $\mathbf{P}'$.

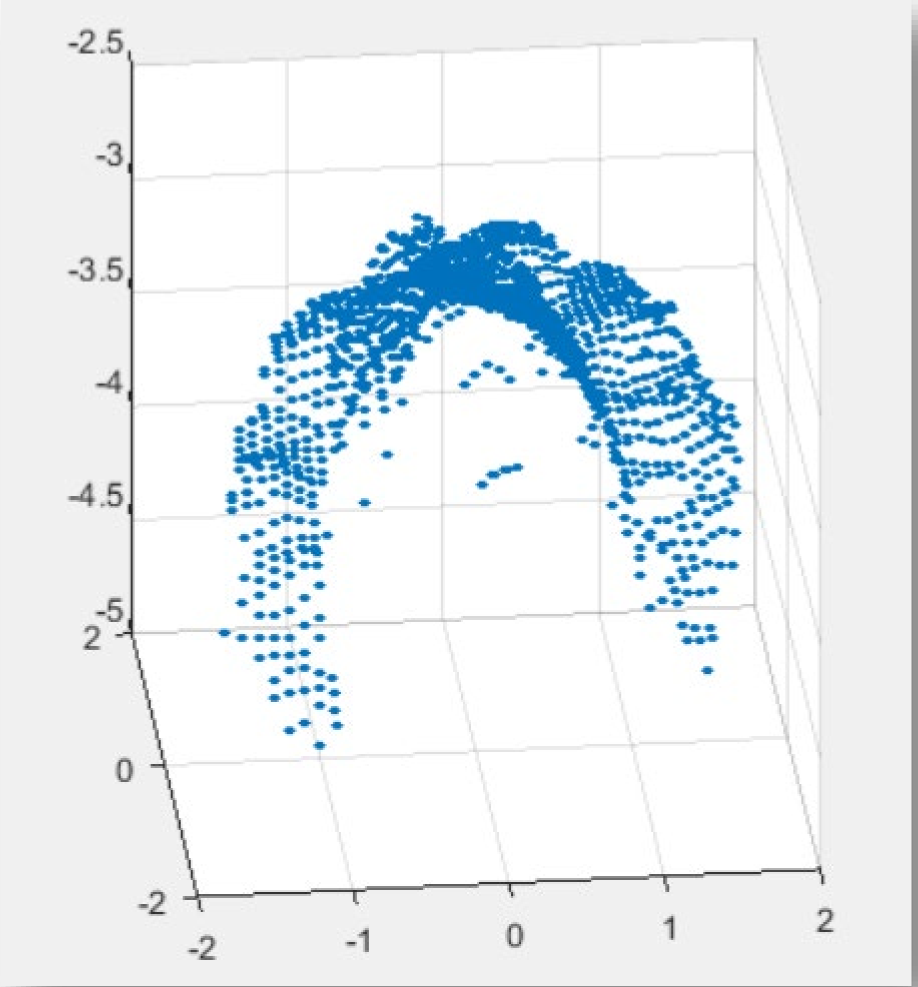
b) Extend your algorithm from exercise 2 for the planar 2D homography to a *spatial* 3D *homography* $\mathbf{H}$. Determine the spatial transformation of the five projective object points $\mathbf{X}_{P2}$ to the corresponding Euclidean object points $\mathbf{X}_E$.

c) Apply this transformation $\mathbf{H}$ to all object points of your projective reconstruction $\mathbf{X}_{P1}$ and visualize the result of the *Euclidean reconstruction* spatially.

Assignment 5 Part 2: *Direct Euclidean reconstruction*

Example Results after **Part 2**





Bauhaus-Universität Weimar

Assignment 5: *Projective and direct Euclidean reconstruction*

Hints for Part 1

- Computation of **F** using all available points (re-use Assignment 4)
- Camera definition from **F** using epipols (Lecture 8)
- First camera normalized (Lecture 8)
- Triangulation:
  - Solve one linear equation system for each point (Lecture 8)
  - → One system for each point (A is a 4x4 matrix)

$$\mathbf{A}\mathbf{X} = \mathbf{0}, \quad \mathbf{A} = \begin{bmatrix} x\mathbf{p}^3 - \mathbf{p}^1 \\ y\mathbf{p}^3 - \mathbf{p}^2 \\ x'\mathbf{p}'^3 - \mathbf{p}'^1 \\ y'\mathbf{p}'^3 - \mathbf{p}'^2 \end{bmatrix}, \quad \mathbf{X} = \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix}$$

where $\mathbf{p}^i$ denotes the row i of $\mathbf{P}$

- Normalize 3D points **X** using **W**

Assignment 5: *Projective and direct Euclidean reconstruction*

Hints for Part 2

- Input for 3D Homography **H**:

  5 given object points (X_E) and the corresponding **5 triangulated points** X_{p2}

- Transformation using **H**: left-side multiplication of X_{p1} (Result of part 1)

  → $X_{result} = H * X_{p1}$

- Normalize X_{result} (homogeneous→Euclidean)

Happy Holidays!

Bauhaus-
Universität
Weimar