Bauhaus-
Universität
Weimar

Photogrammetric Computer Vision

Final Exercise

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	**Essential Matrix Estimation and Non-linear Optimization**

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**	**–**	**23.03.25**

Assignment 6 – sample solution

Assignment 6: *Stereo image matching using* ***normalized cross-correlation***

horizontal scanlines = **epipolar lines**

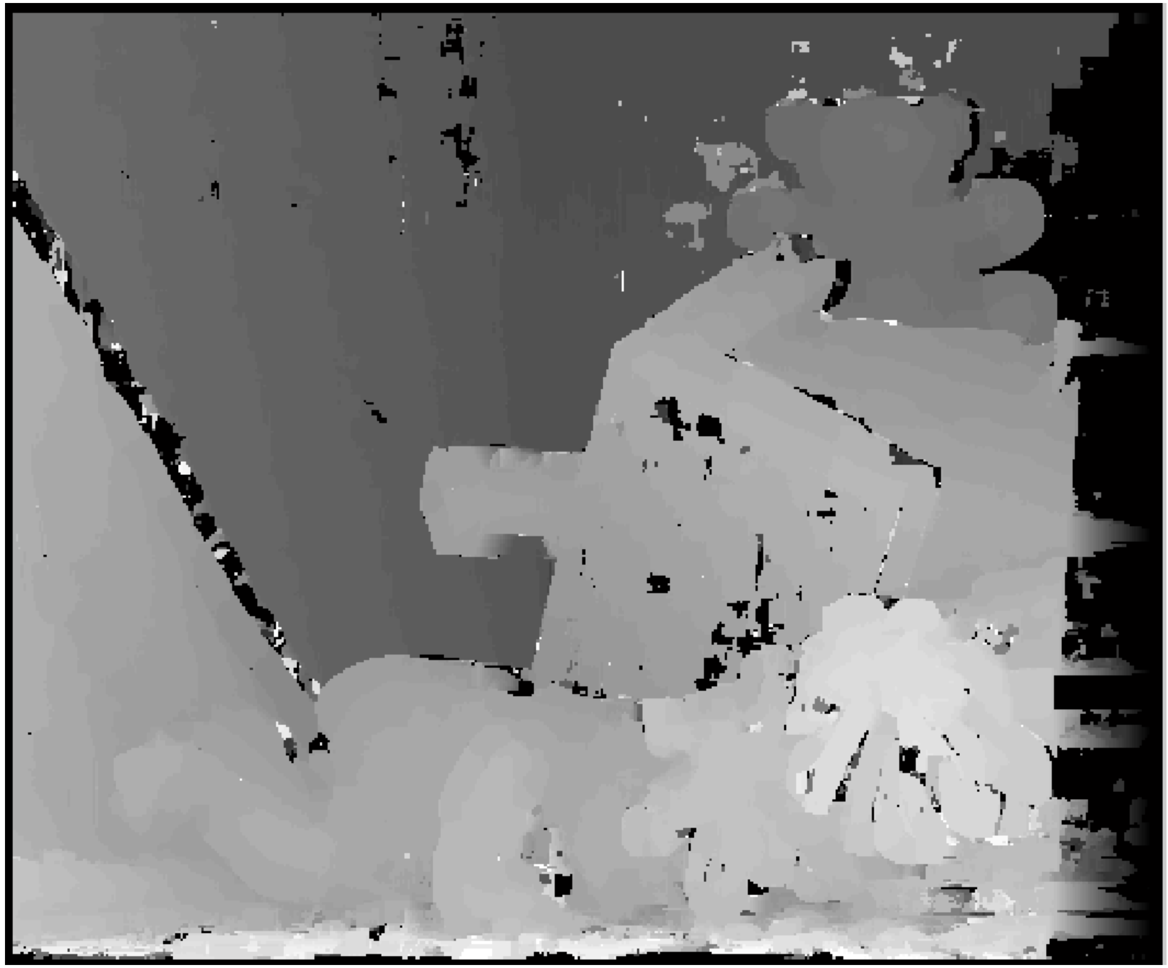
Reference image → left

Search image → right

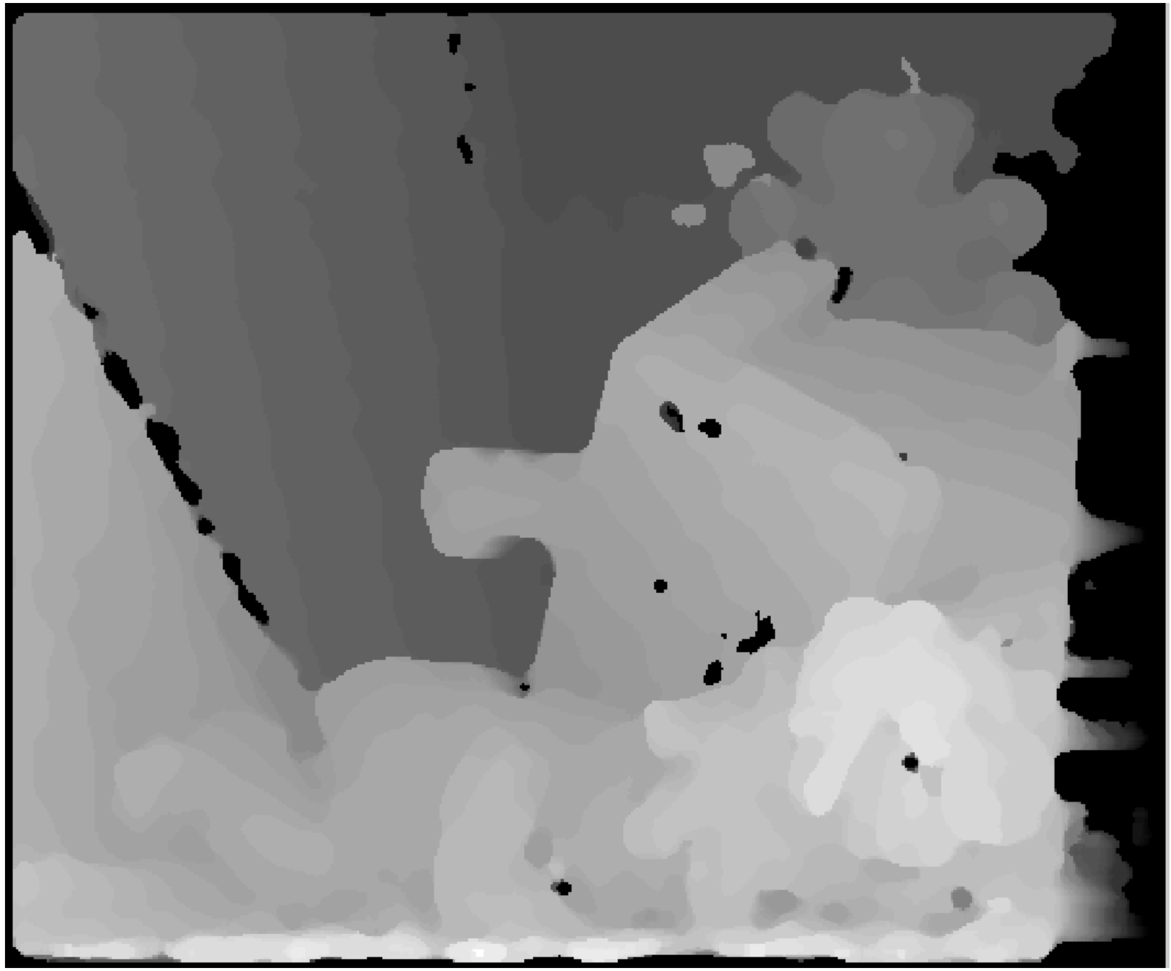
$$\rho_{NCC}(a,b) = \frac{\sigma_{ab}}{\sqrt{\sigma_a^2 \cdot \sigma_b^2}}$$

$$= \frac{\frac{1}{n^2}\left(\sum_{i,j=1}^{n} a(i,j)\cdot b(i,j)\right) - \bar{a}\cdot\bar{b}}{\sqrt{\left(\frac{1}{n^2}\left(\sum_{i,j=1}^{n} a(i,j)^2\right) - \bar{a}^2\right)\cdot\left(\frac{1}{n^2}\left(\sum_{i,j=1}^{n} b(i,j)^2\right) - \bar{b}^2\right)}}$$

Assignment 6: Sample results



Depth Map



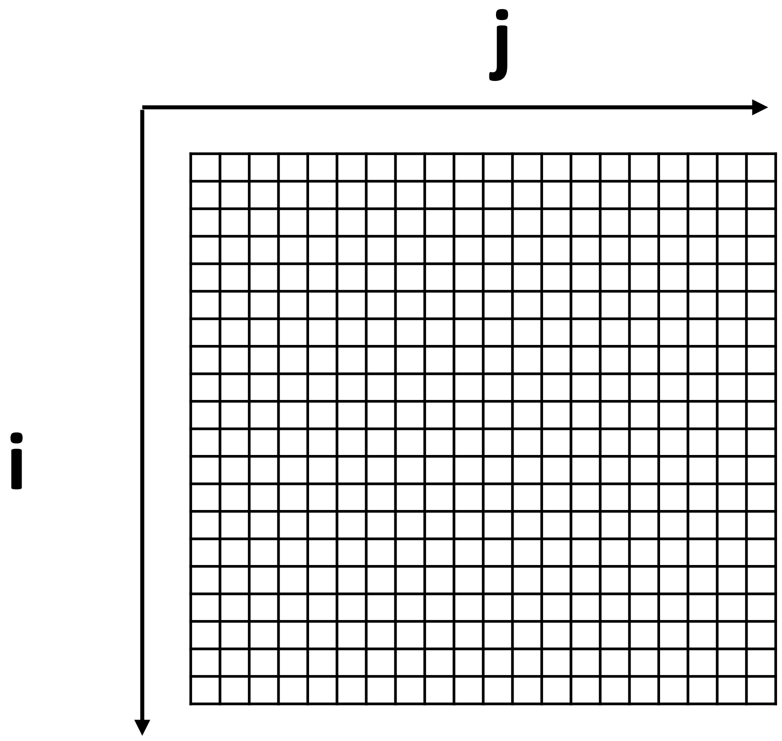
Smoothed Depth Map

Bauhaus-Universität Weimar

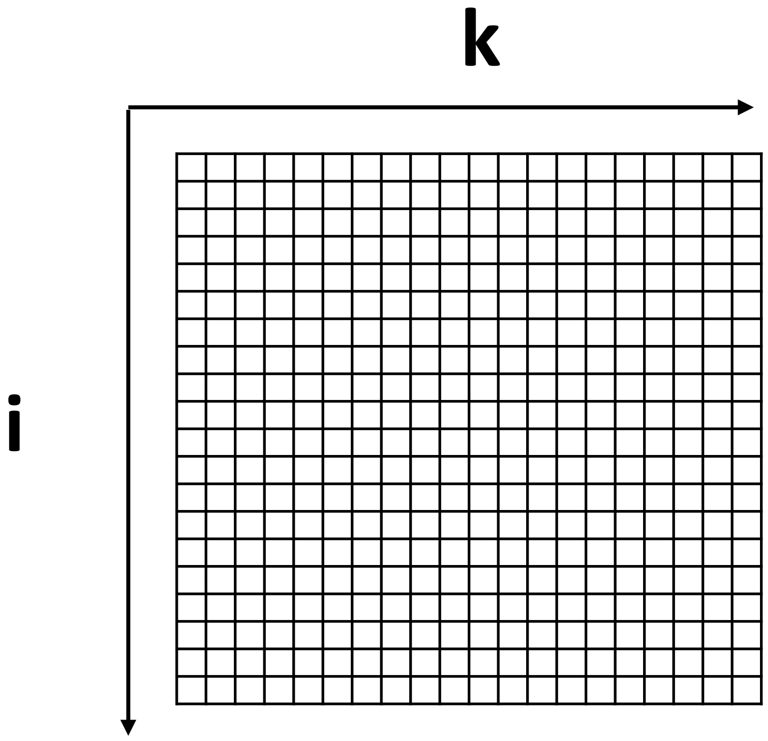
Sample code: Part 1

```
function exercise6
%        =========
r = 2;                                          % Image window radius (1...)
thres = 0.5;                               % Threshold for correlation (-1..1)
dmin = 10; dmax = 50;                         % Minimum and maximum disparity
left  = double(imread('left.png'));              % Read stereo normal images
right = double(imread('right.png'));
[h, w] = size(left);                            % Left image is the reference
D = zeros(h, w);                                   % Initialize disparity map

[lm, lms] = precalc(left, r);                   % Pre-calculate the mean and
[rm, rms] = precalc(right, r);                        % the mean of squares
...


function [m, ms] = precalc(img, r)           % Acceleration by pre-calculation
%        ========================
[h, w] = size(img);                                            % Image size
m  = zeros(h, w);                               % Initialize result matrices
ms = zeros(h, w);
for i = 1+r : h-r                               % For each row i and column j
    for j = 1+r : w-r                        % with a distance r to the border
        A = img(i-r : i+r, j-r : j+r);              % Define an image window A
        m(i, j)  = mean2(A);                                       % Mean of A
        ms(i, j) = mean2(A.*A);                              % Mean of squares
    end
end
```

Assignment 6: *Stereo image matching using* ***normalized cross-correlation***



Reference image
→ left



Search image
→ right

Precomputed value

Sample code: Part 2

```
function exercise6
%        =========
%...
for i = 1+r : h-r                        % For each row i and column j of the reference
    for j = 1+r : w-r                        % image in a distance r from the border
        cmax = thres;
        start = min(j+dmin, w-r);                       % Crop the search space j+dmin
        stop  = min(j+dmax, w-r);                      % to j+dmax at the right border
        A = left(i-r : i+r, j-r : j+r);                    % Define reference window A
        vl = lms(i, j) - lm(i, j)^2;                                  % Variance of A
        if vl > 0                                              % If A contains texture
            for k = start : stop
                B = right(i-r : i+r, k-r : k+r);                  % Search window B
                vr = rms(i, k) - rm(i, k)^2;                          % Variance of B
                if vr > 0                     % If B contains texture calculate NCC
                    cc = (mean2(A.*B) - lm(i, j) * rm(i, k)) / sqrt(vl * vr);
                    if cc > cmax                   % Maximize correlation coefficient
                        cmax = cc;                                % Winner takes all
                        D(i, j) = k - j;                     % Store column difference
                    end
                end
            end
        end
    end
end
figure(2); imshow(D, []);                    % Show disparities as gray value image
```

Sample code: Part 2

Precomputed value

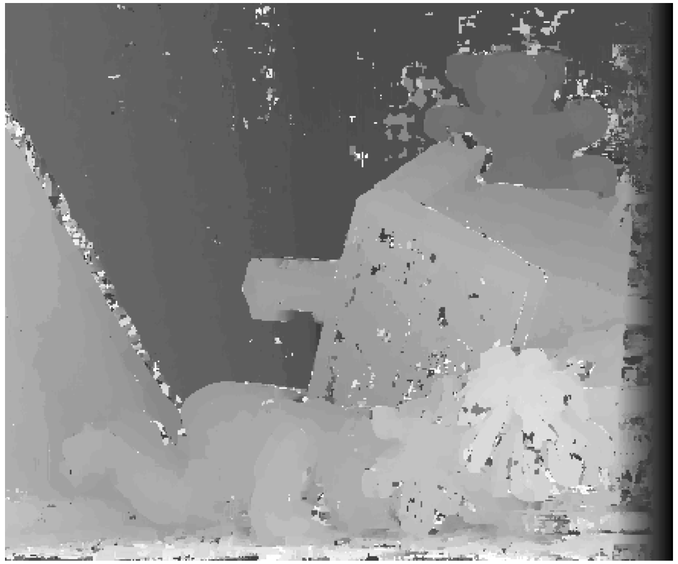
```
function exercise6
%        =========
%...
for i = 1+r : h-r                      % For each row i and column j of the reference
    for j = 1+r : w-r                        % image in a distance r from the border
        cmax = thres;
        start = min(j+dmin, w-r);                        % Crop the search space j+dmin
        stop  = min(j+dmax, w-r);                        % to j+dmax at the right border
        A = left(i-r : i+r, j-r : j+r);                     % Define reference window A
        vl = lms(i, j) - lm(i, j)^2;                                    % Variance of A
        if vl > 0                                                 % If A contains texture
            for k = start : stop
                B = right(i-r : i+r, k-r : k+r);                     % Search window B
                vr = rms(i, k) - rm(i, k)^2;                                % Variance of B
                if vr > 0                         % If B contains texture calculate NCC
                    cc = (mean2(A.*B) - lm(i, j) * rm(i, k)) / sqrt(vl * vr);
                    if cc > cmax                      % Maximize correlation coefficient
                        cmax = cc;                                    % Winner takes all
                        D(i, j) = k - j;                         % Store column difference
                    end
                end
            end
        end
    end
end
D = medfilt2(D, [9, 9]);               % Optional: Median filtering of the result
figure(2); imshow(D, []);                    % Show disparities as gray value image
```

Bauhaus-Universität Weimar

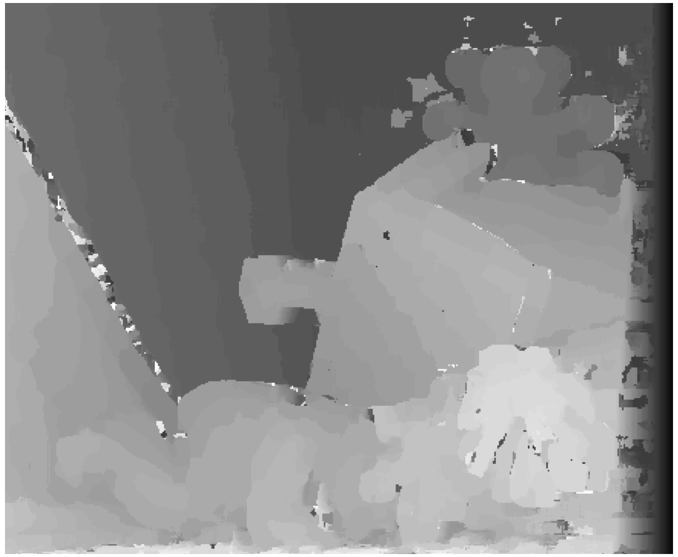
Assignment 6: Window size influence

$d_{min} = 12,$
$d_{max} = 50$

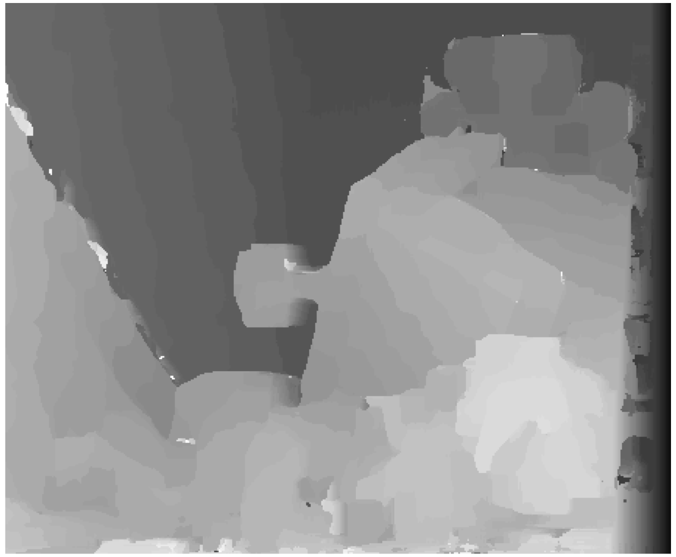
$r = 1$



$r = 3$



$r = 5$



$r = 10$

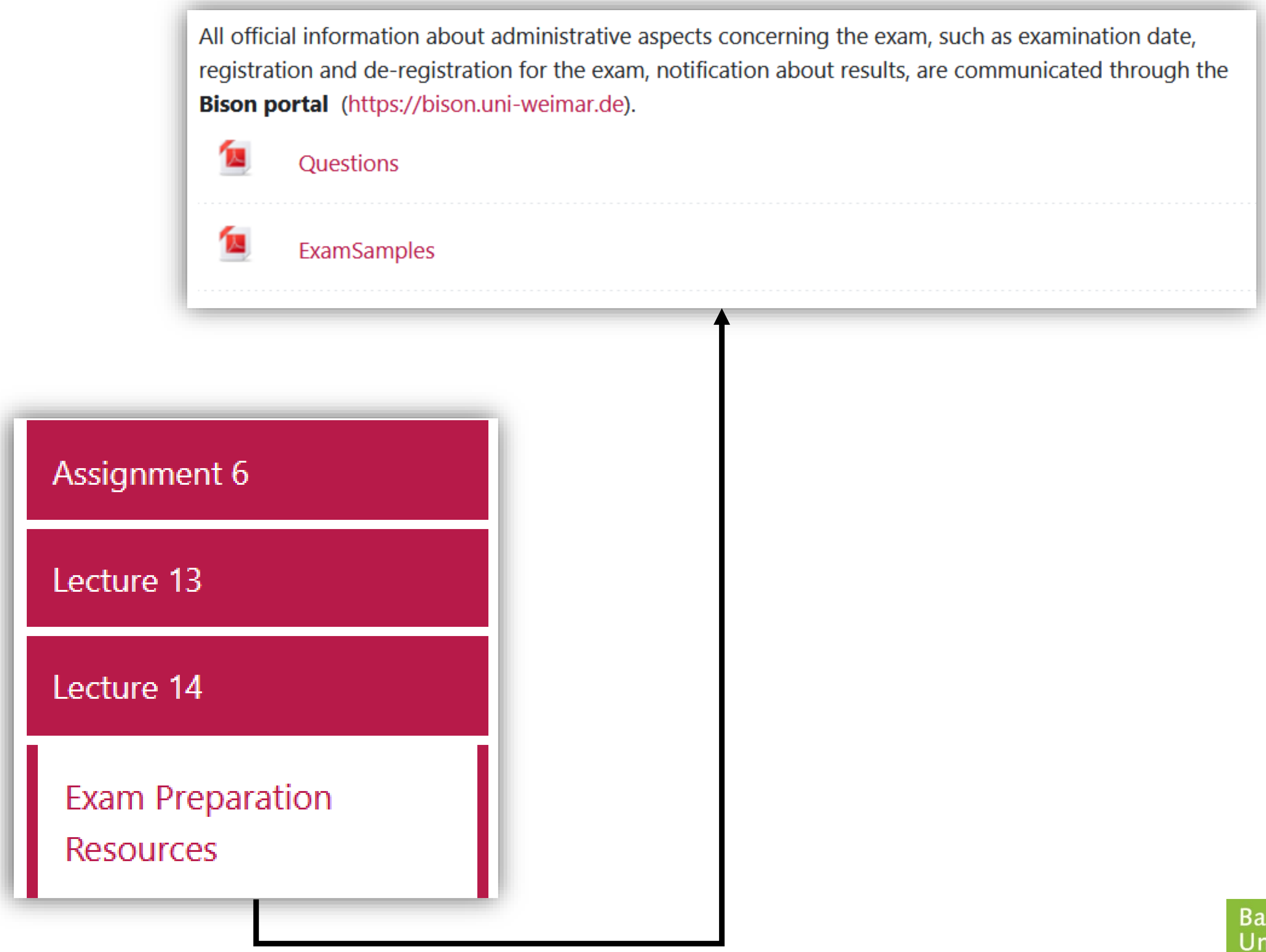
Assignment 6: Result for swapped images

Exam Information

- Date: 17th of February 2025 (Monday)
- Staring time: 13:30
- Place: Maurice-Halbwachs-Auditorium (former Audimax) + Lecture Hall A
- Duration: 90 minutes (plus some additional time for initial instructions)
- Auxiliary resources: None
- Also good to know:
  - We will provide you with paper to write on!
  - The use of calculators will NOT be allowed! (You won’t be needing such either.)
  - Bring your student ID (THOSKA)
- Preparation material:
  - Lecture slides and (old) videos
  - List of questions
  - Old exam samples

Exam Preparation

- Try solving all **old exam samples**
- Revisit old **lecture videos**
- Form **study groups** and:
  - discuss your solutions
  - try formulating new possible exam questions for each other
  - actively explain to each other core topics form the lecture content



Bauhaus-Universität Weimar

Photogrammetric Computer Vision

Final Project

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

Final Project

- Topic: **Essential Matrix Estimation and Non-linear Optimization**
- Submission deadline: **23.03 2025, 23:00**
- Work in small groups up to 4 people
  - generally study groups should stay the same
  - new groups may be formed only upon explicit request
- Submission
  - upload only via Moodle
  - single submission per group
  - source code + short documentation

Final Project

- Topic: **Essential Matrix Estimation and Non-linear Optimization**
- Submission deadline: **23.03 2025, 23:00**
- Additional Requirements
  - include full **names** and **student IDs** of all group members in the project documentation
  - **comment** your source code (e. g. all major implementation steps should be indicated)
  - use meaningful variable names
  - aim for **short** (no more than 5 pages) and clear documentation
  - **reference** all used external **sources** (e. g. lecture slides from other universities, reference implementations, etc.)
  - work independently within your own group

Project Description

Task 1: **Essential Matrix Estimation**

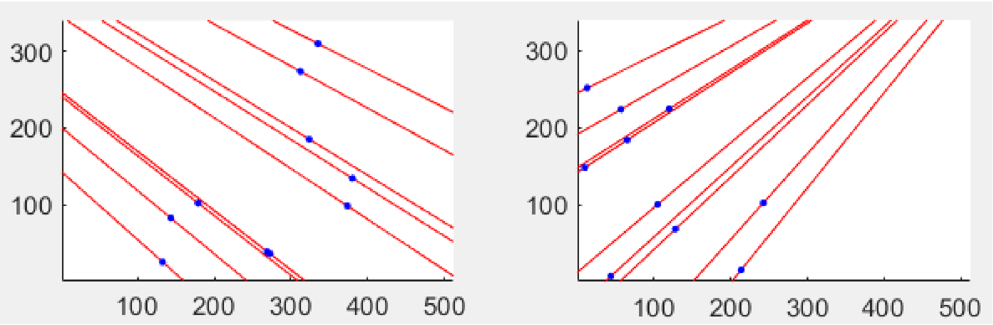
- Use the provided image and object points

a) compute calibration matrices K1 and K2

b) estimation essential matrix E

c) resolve the fourfold ambiguity

d) compute and visualize epipolar lines



Task 2: **Non-linear Optimization**

- Use the output results from Task 1

a) compute the geometric error

b) perform a non-linear optimization

c) reevaluate the geometric error after performing the optimization step