Photogrammetric Computer Vision

Exercise 3

Winter semester 21/22

(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 -

***Depending on the regulations of your study program, this project might be optional for you!**

If you are not sure about the exact requirements for your study program,

please consult with a representative of the Academic Affairs Office in charge!

Agenda

	Beginning:	Submission deadline:
Assignment 1.	~~18.10.21~~	~~31.10.21~~
Assignment 2.	~~01.11.21~~	~~14.11.21~~
Assignment 3.	**15.11.21**	**28.11.21**
Assignment 4.	29.11.21	12.12.21
Assignment 5.	13.12.21	09.01.22
Assignment 6.	10.01.22	23.01.22
Final Project. *	24.01.22	- will be announced later -

***Depending on the regulations of your study program, this project might be optional for you!**

If you are not sure about the exact requirements for your study program,
please consult with a representative of the Academic Affairs Office in charge!

Assignment 2 – sample solution

Sample code 1/2

```
% Image mosaicking using general projective 2D transformations (homographies)

function exercise2
%        =========
f = imread('image1.ppm');                          % Read three images f, g, h
g = imread('image2.ppm');
h = imread('image3.ppm');                   % Adjust image f to image g and then
i = mosaic_image(h, mosaic_image(f, g));          % image h to the combined image
imshow(i);                                                   % Show mosaic result
```

```
function i = mosaic_image(f, g)                              % Combine two images
%          ==================
imshow(f); x1 = get_points;   % Show image f and g and get 4 homologous points
imshow(g); x2 = get_points;                       % in image f and then in image g
H = homography2(x1, x2);                                   % Compute homography H
i = geokor(H, f, g);          % Rectify image f using H and combine it with g
```

```
function p = get_points           % Measure homogeneous coordinates interactively
%          ==========                                      | x1, x2,     xn |
p = []; but = 1;                                       % p = | y1, y2, ... yn |
while but == 1                                         %     | 1 , 1 ,     1  |
    [x, y, but] = ginput(1);
    if but == 1
        p = [p [x y 1]'];
        hold on; plot(x, y, 'r+'); hold off;
    end
end
```

```
function H = homography2(x1, x2)                  % Planar projective transformation
%          ===================
T1 = condition2(x1); c1 = T1 * x1;                        % Image point conditioning
T2 = condition2(x2); c2 = T2 * x2;
A = design_homo2(c1, c2);                                     % Build design matrix
h = solve_dlt(A);                                    % Linear least squares solution
H = inv(T2) * reshape(h, 3, 3)' * T1;    % Reshape row-wise and deconditioning
```

Sample code 1/2

```
% Image mosaicking using general projective 2D transformations (homographies)

function exercise2
%        =========
f = imread('image1.ppm');                           % Read three images f, g, h
g = imread('image2.ppm');
h = imread('image3.ppm');                 % Adjust image f to image g and then
i = mosaic_image(h, mosaic_image(f, g));       % image h to the combined image
imshow(i);                                               % Show mosaic result
```

```
function i = mosaic_image(f, g)                          % Combine two images
%            ===================
imshow(f); x1 = get_points;     % Show image f and g and get 4 homologous points
imshow(g); x2 = get_points;                   % in image f and then in image g
H = homography2(x1, x2);                               % Compute homography H
i = geokor(H, f, g);           % Rectify image f using H and combine it with g
```

```
function p = get_points         % Measure homogeneous coordinates interactively
%            ==========                                 | x1, x2,      xn |
p = []; but = 1;                                  % p = | y1, y2, ... yn |
while but == 1                                    %     | 1 , 1 ,      1  |
    [x, y, but] = ginput(1);
    if but == 1
        p = [p [x y 1]'];
        hold on; plot(x, y, 'r+'); hold off;
    end
end
```

```
function H = homography2(x1, x2)                % Planar projective transformation
%            ====================
T1 = condition2(x1); c1 = T1 * x1;                    % Image point conditioning
T2 = condition2(x2); c2 = T2 * x2;
A = design_homo2(c1, c2);                                % Build design matrix
h = solve_dlt(A);                                % Linear least squares solution
H = inv(T2) * reshape(h, 3, 3)' * T1;     % Reshape row-wise and deconditioning
```

Sample code 2/2

```
function H = homography2(x1, x2)                  % Planar projective transformation
%            ===================
T1 = condition2(x1); c1 = T1 * x1;                       % Image point conditioning
T2 = condition2(x2); c2 = T2 * x2;
A = design_homo2(c1, c2);                                     % Build design matrix
h = solve_dlt(A);                                      % Linear least squares solution
H = inv(T2) * reshape(h, 3, 3)' * T1;      % Reshape row-wise and deconditioning
```

```
function T = condition2(x)                     % Conditioning matrix 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 A = design_homo2(x1, x2)                 % Design matrix for 2D homography
%            ====================
A = [];
for i = 1 : size(x1, 2)
    A = [ A; -x2(3, i)*x1(:, i)'  0   0   0  x2(1, i)*x1(:, i)' ;
             0   0   0  -x2(3, i)*x1(:, i)'  x2(2, i)*x1(:, i)' ];
end
```

```
function x = solve_dlt(A)   % Direct linear transformation, solver for A*x = 0
%            ============
[U, D, V] = svd(A);                                 % Singular value decomposition
x = V(:, end); % Last column is singular vector to the smallest singular value
```

Bauhaus-Universität Weimar

Sample code 2/2

```
function H = homography2(x1, x2)          % Planar projective transformation
%         ====================
T1 = condition2(x1); c1 = T1 * x1;                % Image point conditioning
T2 = condition2(x2); c2 = T2 * x2;
A = design_homo2(c1, c2);                              % Build design matrix
h = solve_dlt(A);                               % Linear least squares solution
H = inv(T2) * reshape(h, 3, 3)' * T1;    % Reshape row-wise and deconditioning
```

column-wise: reshape(p, 3, 3)

≠

row-wise: reshape(p, 3, 3)'

```
>> p = [1:9]

p =

     1     2     3     4     5     6     7     8     9

>> p_mat = reshape(p, 3, 3)

p_mat =

     1     4     7
     2     5     8
     3     6     9

>> p_mat2 = reshape(p, 3, 3)'

p_mat2 =

     1     2     3
     4     5     6
     7     8     9
```

Sample code 2/2

```
function H = homography2(x1, x2)            % Planar projective transformation
%            ====================
T1 = condition2(x1); c1 = T1 * x1;                  % Image point conditioning
T2 = condition2(x2); c2 = T2 * x2;
A = design_homo2(c1, c2);                                % Build design matrix
h = solve_dlt(A);                                 % Linear least squares solution
H = inv(T2) * reshape(h, 3, 3)' * T1;     % Reshape row-wise and deconditioning
```

```
function T = condition2(x)                 % Conditioning matrix 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 A = design_homo2(x1, x2)            % Design matrix for 2D homography
%            ====================
A = [];
for i = 1 : size(x1, 2)
    A = [ A; -x2(3, i)*x1(:, i)'  0   0   0  x2(1, i)*x1(:, i)' ;
               0   0   0  -x2(3, i)*x1(:, i)'  x2(2, i)*x1(:, i)' ];
end
```

```
function x = solve_dlt(A)   % Direct linear transformation, solver for A*x = 0
%            ============
[U, D, V] = svd(A);                               % Singular value decomposition
x = V(:, end); % Last column is singular vector to the smallest singular value
```

Assignment 2 – example results

Assignment 2 – example results

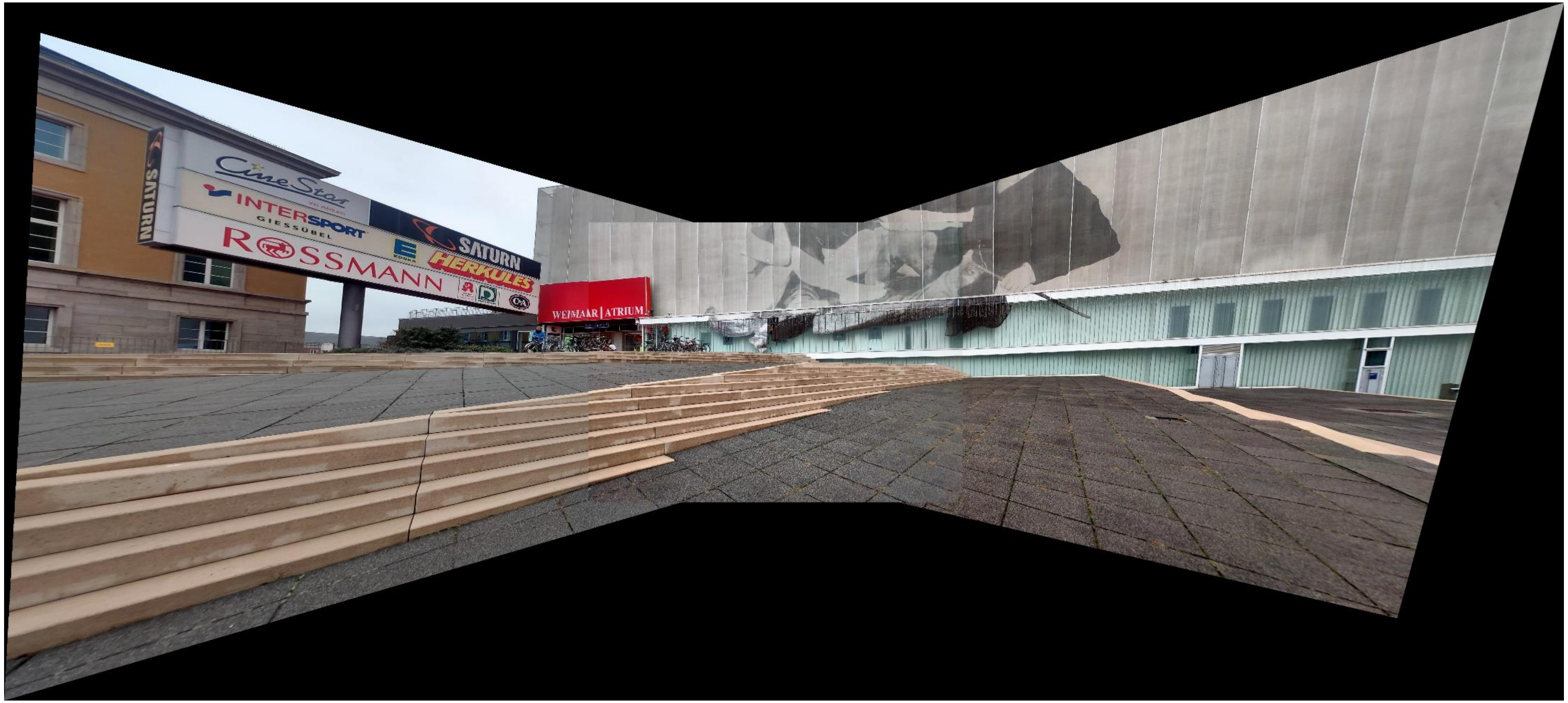
Assignment 2 – example results

Assignment 2 – example results

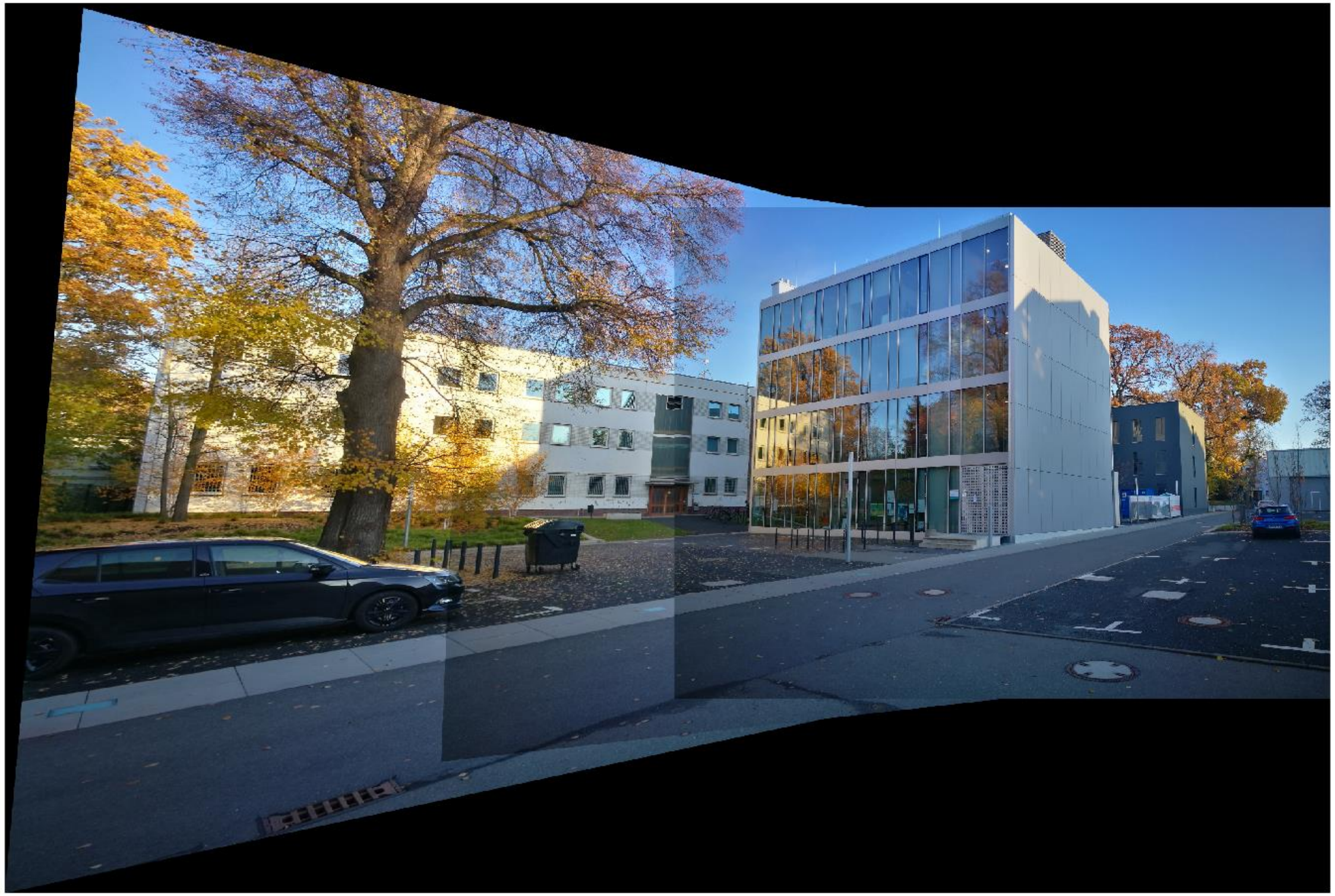
Bauhaus-
Universität
Weimar

Assignment 2 – example results

Assignment 2 – example results



Assignment 2 – example results



Bauhaus-
Universität
Weimar

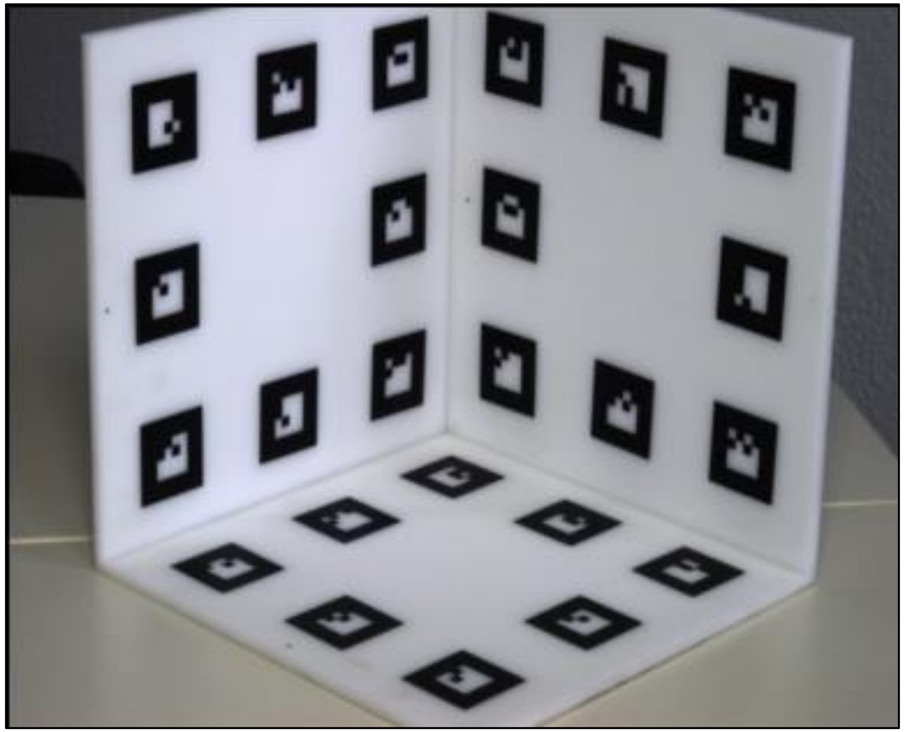
Assignment 3: *Camera calibration using DLT*

Bauhaus-
Universität
Weimar

Assignment 3: *Camera calibration using DLT*

1) Image acquisition

   - one picture of an appropriate calibration object
   - brief description of the chosen calibration object
   - technical information about the used camera



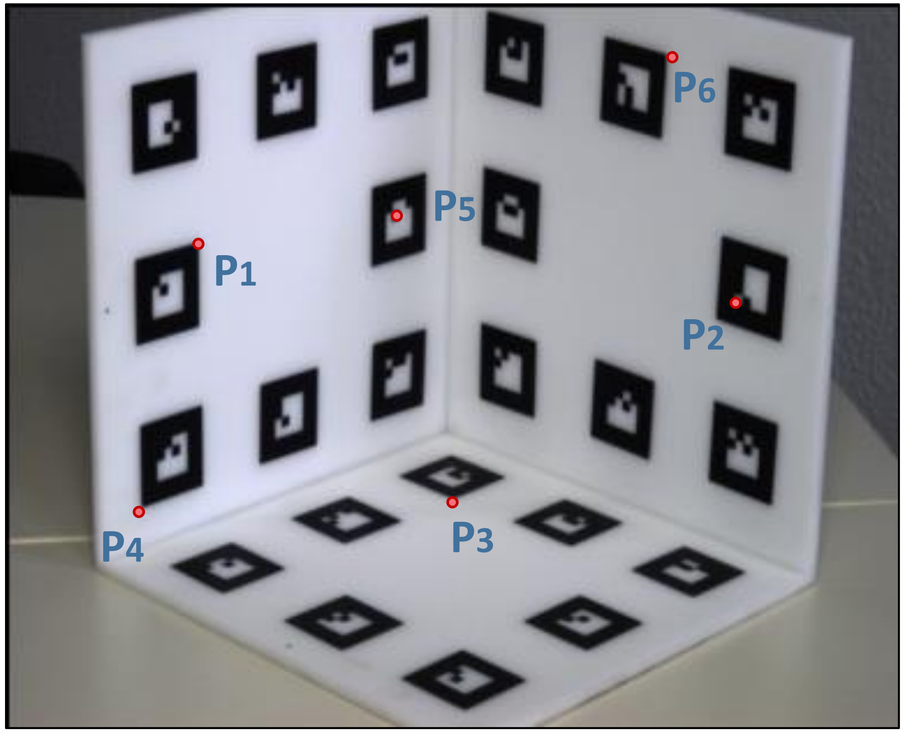
Assignment 3: *Camera calibration using DLT*

1) Image acquisition

   - one picture of an appropriate calibration object
   - brief description of the chosen calibration object
   - technical information about the used camera

2) Control point measurements

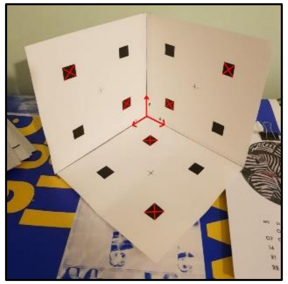
   - object coordinates of at least 6 control points
   - axes of the object coordinate system
   - measurements precision

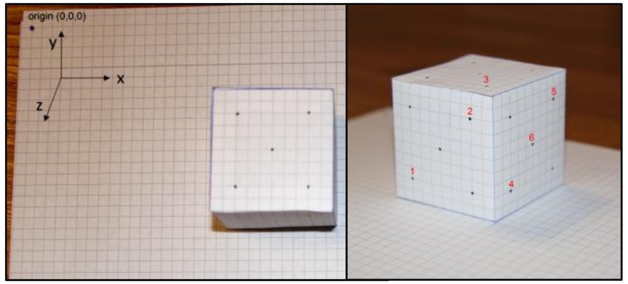


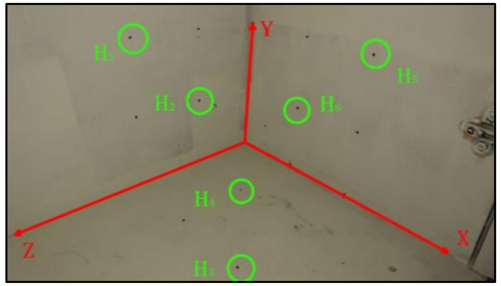
Assignment 3: *Camera calibration using DLT*

1) Image acquisition
   - one picture of an appropriate calibration object
   - brief description of the chosen calibration object
   - technical information about the used camera

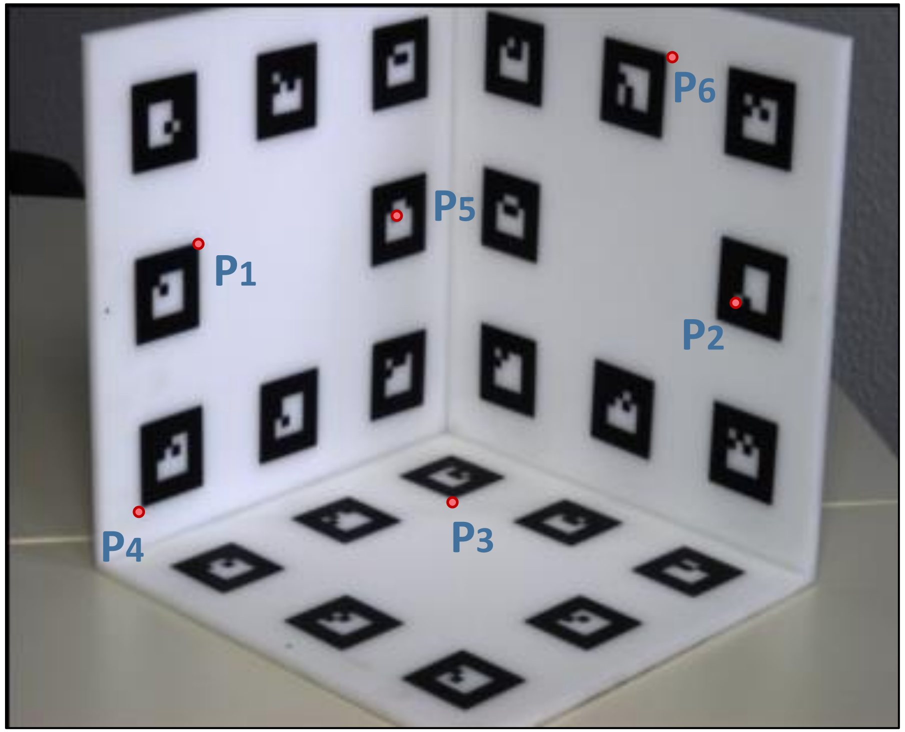
2) Control point measurements
   - object coordinates of at least 6 control points
   - axes of the object coordinate system
   - measurements precision







examples of annotated calibration objects from previous students submissions



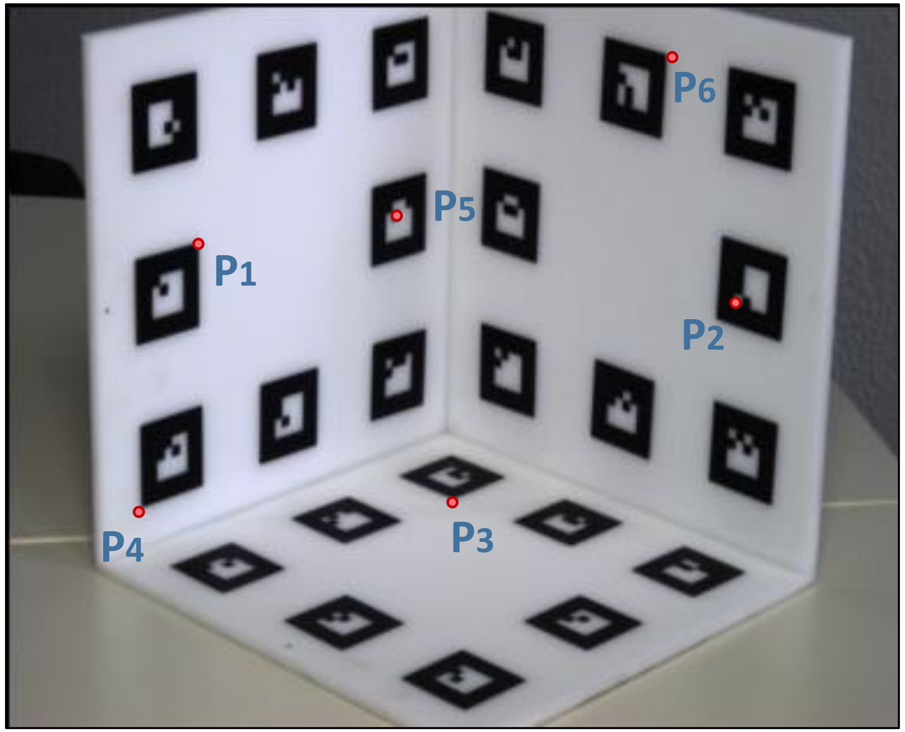
Bauhaus-Universität Weimar

Assignment 3: *Camera calibration using DLT*

1) Image acquisition
   - one picture of an appropriate calibration object
   - brief description of the chosen calibration object
   - technical information about the used camera

2) Control point measurements
   - object coordinates of at least 6 control points
   - axes of the object coordinate system
   - measurements precision

3) Computation of the projection matrix
   - spatial resection using DLT with help of SVD



Assignment 3: *Camera calibration using DLT*

1) Image acquisition

   - one picture of an appropriate calibration object
   - brief description of the chosen calibration object
   - technical information about the used camera

2) Control point measurements

   - object coordinates of at least 6 control points
   - axes of the object coordinate system
   - measurements precision

3) Computation of the projection matrix

   - spatial resection using DLT with help of SVD

4) Interpretation of the projection matrix

   - factorization of projection matrix (RQD)
   - meaning of extracted parameters
   - quality assessment