Bauhaus-
Universität
Weimar

Photogrammetric Computer Vision
Sample Solution for Assignment 4

```
% Relative orientation of an image pair

function exercise4
%        =========
f = imread('image1.jpg');                                    % Read image pair
g = imread('image2.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
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 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 and reverse conditioning
```

```
function A = design_fund(x1, x2)    % Design matrix for the fundamental 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
```

```
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 = mean(num ./ den)                            % Average epipolar distance
```

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