Exercise 4 - Model Fitting

Computer Vision

Alberto Montes (malberto@student.ethz.ch) November 10, 2016

Line Fitting with RANSAC

For the first exercise which consist to code the RANSAC algorithm for line fitting. The algorithm follows the guidelines of its design, which iterating over a fixed number of times, two points are randomly chosen, fit a line on it and compute the inline and outlines points. If there is enough inline points, the line is refitted for all this points and the one with the best inline ratio is kept to be returned. The code is on the attached code and the results obtained with the examples cloud point are on Figure 1. As the algorithm was design to be less sensitive to outliers, at the results can be seen how the predicted line by the algorithm is almost the same as the original one.

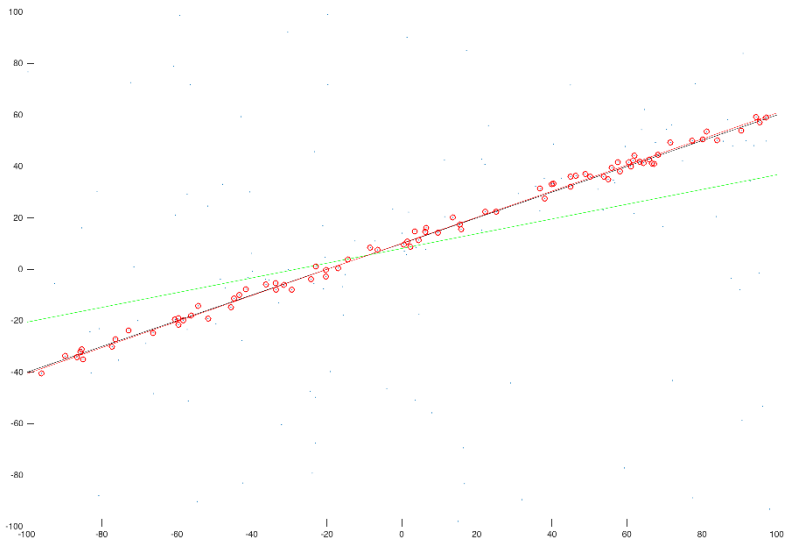

Figure 1: Results of line fitting using RANSAC algorithm.

Fundamental Matrix

The next exercise consist on finding the fundamental matrix from the given points of two different images from the same scenario. The implemented code can be found in the Listing 1.

```
% Compute the fundamental matrix using the eight point algorithm
% Input
%   x1s, x2s     Point correspondences
%
% Output
%   Fh           Fundamental matrix with the det F = 0 constraint
%   F            Initial fundamental matrix obtained from the eight point
    algorithm
%
function [Fh, F] = fundamentalMatrix(x1s, x2s)
```

```
10
11      [x1n, T1] = normalizePoints2d(x1s);
12      [x2n, T2] = normalizePoints2d(x2s);
13
14      W = [ repmat(x2n(1,:)',1,3) .* x1n', repmat(x2n(2,:)',1,3) .* x1n',
        x1n(1:3,:)'];
15      [~, ~, V] = svd(W);
16      F = reshape(V(:,end),3,3)';
17
18      [u, s, v] = svd(F);
19      Fh = T2' * u * diag([s(1) s(5) 0]) * v' * T1;
20  end
```

Listing 1: fundamentalMatrix.m

The implementation first normalize all the given points. Then with the eight-point algorithm find the values of the non-singular fundamental matrix F, and then, enforcing the singularity constrain, finds the fundamental matrix $\hat{F}$. Once is computed the fundamental matrix, the epipolar lines can be drawn from one image point to the other one, an on Figure 2 there is the result.

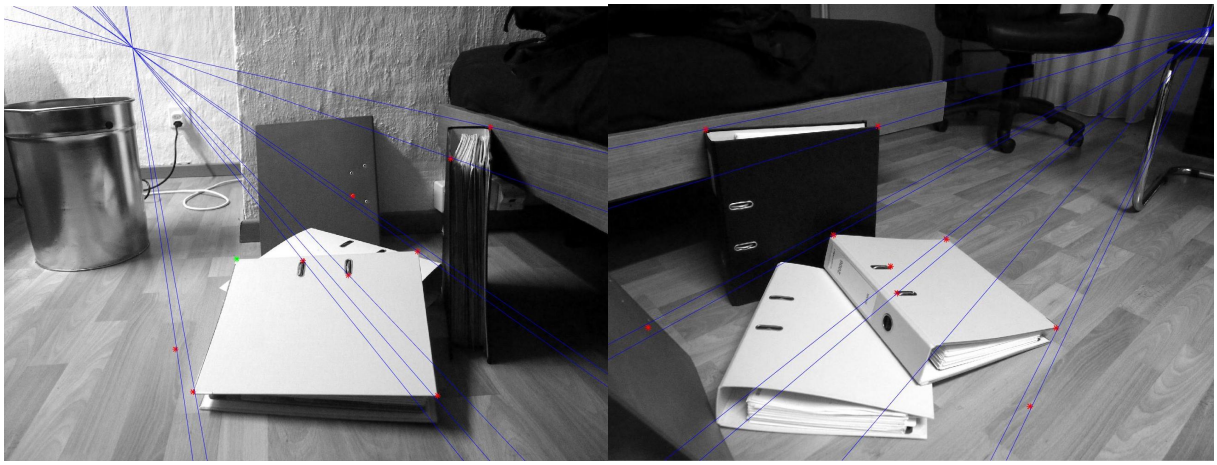

Figure 2: Epipolar lines obtained with the Fundamental Matrix $\hat{F}$.

Essential Matrix

The next step, has been to compute the Essential Matrix from the normalized points using the camera matrix. The way to compute it is the same as the eight-point algorithm previously used with the fundamental matrix. To take into account, it has been enforced the restriction to Essential Matrix to have the same value at the first two singular values. The implementation is on Listing 2 and the epipolar lines drawn from the computed essential matrix is plot on Figure 3.

```
1  % Compute the essential matrix using the eight point algorithm
2  % Input
3  %   x1s, x2s     Point correspondences 3xn matrices
4  %
5  % Output
6  %   Eh           Essential matrix with the det E = 0 constraint and the
       constraint that the first two singular values are equal
7  %   E            Initial essential matrix obtained from the eight point
       algorithm
8  %
9
10 function [Eh, E] = essentialMatrix(x1s, x2s)
11
```

```
    W = [ repmat(x2s(1,:)',1,3) .* x1s', repmat(x2s(2,:)',1,3) .* x1s',
    x1s(1:3,:)'];
    [~, ~, V] = svd(W);
    E = reshape(V(:,end),3,3)';

    [u, s, v] = svd(E);
    r = s(1); s = s(5);
    Eh = u * diag([(r+s)/2, (r+s)/2, 0]) * v';

end
```

Listing 2: essentialMatrix.m

Figure 3: Epipolar lines obtained with the Essential Matrix E.

Camera Matrix

To compute the camera matrix P', it has been necessary to compute decompose the Essential Matrix into R and t and find the better combination of both (two possible directions and two possible rotations). To do so, it has been computed the four combinations, and then chose the one which the projected coordinates are placed in front of the camera, so the z coordinate is positive for all the available points (the 3D points representation is on Figure 4). On Listing 3 is the implementation to decompose the Essential Matrix. In addition I would like to remark that it is important to ensure that the $det(R) = 1$ in positive sign.

```
% Decompose the essential matrix
% Return P = [R|t] which relates the two views
% Yu will need the point correspondences to find the correct solution
    for P
function P = decomposeE(E, x1s, x2s)
    [U,~,V] = svd(E);
    W = [0 -1 0; 1 0 0; 0 0 1];
    R1 = -U*W*V'; % minus sign to enforce det(R)=1
    R2 = -U*W'*V';

    t = U(:,end);
    t = t / norm(t);

    P_c = [eye(3), zeros(3,1)];
    for i = 1:4
        PP = (R1*(i<=2) + R2*(i>2)) * [eye(3), t*((-1)^(i-1))];
        [XS, ~] = linearTriangulation(P_c, x1s, PP, x2s);
        % Check if any of the 3D points is placed behind the camera
```

```
        % if so, P it will not be valid.
        if sum(XS(3,:) < 0) == 0
            P = PP;
            break;
        end
    end
end
```

Listing 3: decomposeE.m

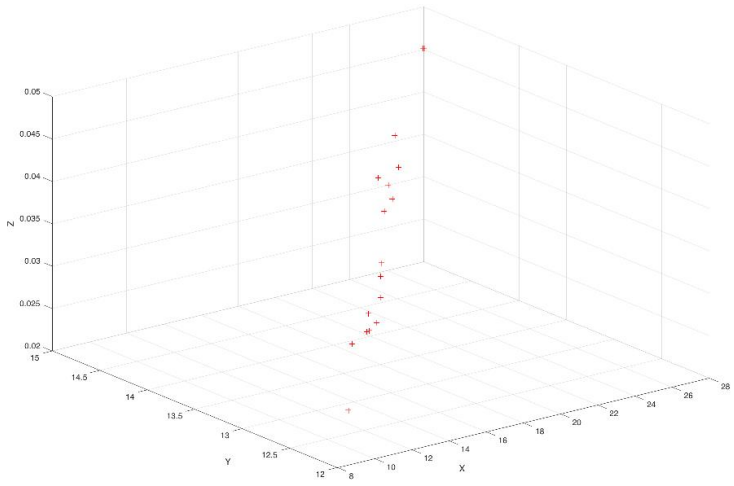

Figure 4: 3D points representation.

Feature Extraction and Matching

To perform a feature matching, it has been computed the SIFT features using the VLEAF implementation and then plot the matching in both formats: side by side as in Figure 5 and overlap as in Figure 6.

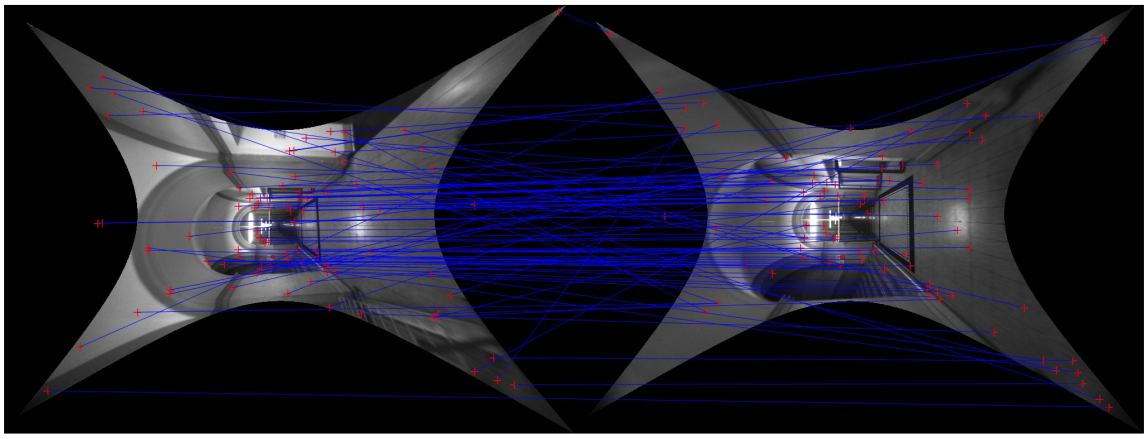

Figure 5: SIFT Features matching.

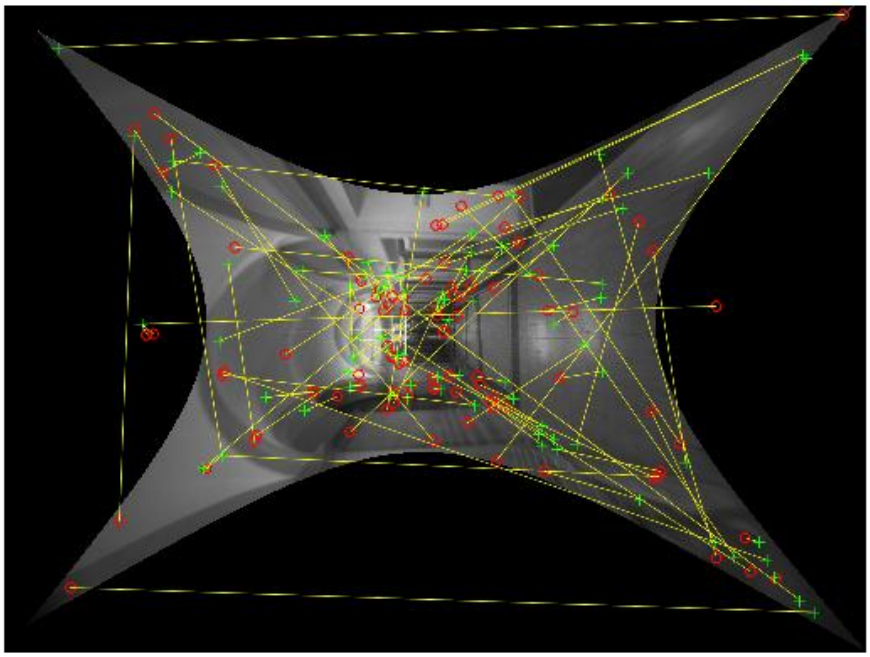

Figure 6: SIFT Features matching (overlap).

8-Point RANSAC

Finally, the 8-point RANSAC algorithm has been used to find the fundamental matrix and improve the features matching seen in Figure 5. The algorithm is the same as in implemented at the beginning, but in this case, the fundamental matrix is computed and the distance is the computation of the distance of the correspondence point to the epipolar line.

On Listing 4 there is the implementation of the algorithm. Also, in addition, on Figure 7 there is plot the inliears matchings as green and the outliers matchings as red.

```
function [in1, in2, out1, out2, m, F] = ransac8pF(xy1, xy2, threshold)

    num = 8;                % 8 points to copute the F matrix
    number = size(xy1,2);   % Total number of points
    bestInNum = 0;          % Best fitting line with largest number of
   inliers
    F = 0;                  % parameters for best fitting line
    in_idx = false(1, number);
    p = .99;

    xy1s = [xy1; ones(1, number)];
    xy2s = [xy2; ones(1, number)];

    for i=1:1000

        idx = randsample(number, num);
        x1s = xy1s(:,idx);
        x2s = xy2s(:,idx);
        [f, ~] = fundamentalMatrix(x1s, x2s);
        inlier_idx = false(1, number);
        for k = 1:number
            err = distPointLine(xy2(:,k), f*xy1s(:,k)) + ...
                  distPointLine(xy1(:,k), f'*xy2s(:,k));
            if err < threshold
                inlier_idx(k) = true;
```

```
            end
        end

        if sum(inlier_idx) > bestInNum
            bestInNum = sum(inlier_idx);
            in_idx = inlier_idx;
            F = f;
        end

    end

    % Recompute the fundamental matrix with all the inliers found
    x1s = xy1(:,in_idx);
    x2s = xy2(:,in_idx);
    [F, ~] = fundamentalMatrix(x1s, x2s);
    % Find one last time all the inliers
    in_idx = false(1, number);
    for k = 1:number
        err = distPointLine(xy2(:,k), f*xy1s(:,k)) + ...
              distPointLine(xy1(:,k), f'*xy2s(:,k));
        if err < threshold
            in_idx(k) = true;
        end
    end

    in1 = xy1(:,in_idx);
    in2 = xy2(:,in_idx);
    out1 = xy1(:,~in_idx);
    out2 = xy2(:,~in_idx);
    m = i;

end
```

Listing 4: ransac8pF.m

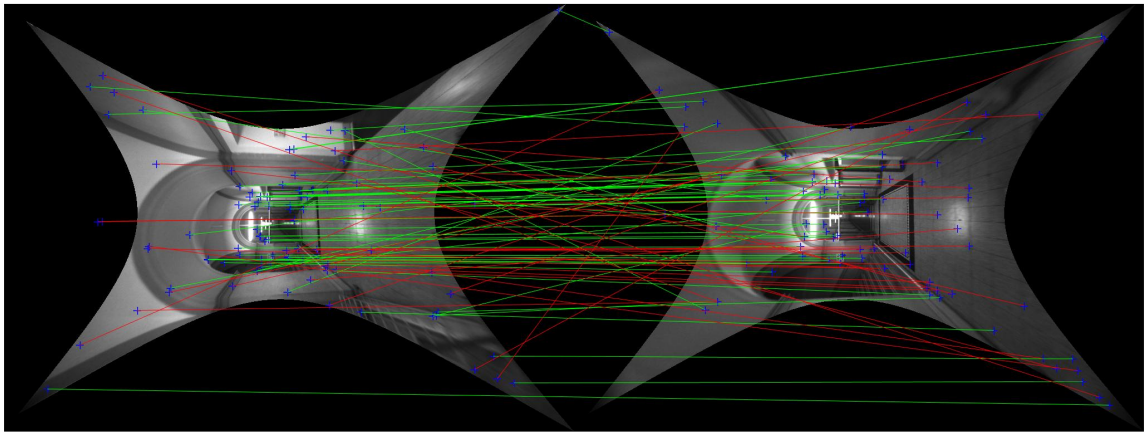

Figure 7: Features matching using the 8-point RANSAC algorithm for Fundamental Matrix