ME491(B)
Homework 5 - Fundamental Matrix estimation

Herman Kolstad Jakobsen
20196493

May 12, 2020

Contents

1 Introduction

The fundamental matrix $\mathbf{F}$ is a relationship between any two images of the same scene that constraints where the projection of points from the scenes can occur in both images. Given the projection of a scene point into one of the images the corresponding point in the other image is constrained to a line, helping the search, and allowing for the detection of wrong correspondences. The relation between corresponding image points, which the fundamental matrix represents, is referred to as epipolar constraint. [1]

In this homework, the fundamental matrix relating two photos taken at KAIST will be estimated. The procedure of estimating the matrix will first be described, before presenting the results. The MATLAB code used to solve the assignment is shown in the last section.

2 Procedure

For any pair of matching points $\mathbf{x} \leftrightarrow \mathbf{x}'$ in two images, the fundamental matrix $\mathbf{F}$ fulfills the following equation

$$\mathbf{x}'^T \mathbf{F} \mathbf{x} = 0 \tag{1}$$

Hence, eq. (1) can be used to compute the unknown matrix $\mathbf{F}$. Representing the image points as homogeneous coordinates $\mathbf{x} = (x, y, 1)^T$ and $\mathbf{x}' = (x', y', 1)^T$, eq. (1) can then be written out as

$$x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0 \tag{2}$$

This equation can be further rearranged to a vector inner product

$$(x'x, x'y, x', y'x, y'y, y', x, y, 1)\mathbf{f} = 0 \tag{3}$$

where $\mathbf{f}$ is a 9-vector made up of the entries of $\mathbf{F}$ in row-major order. A set of n point matches will then result in the following set of linear equations

$$\mathbf{Af} = \begin{bmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{bmatrix} \mathbf{f} = 0 \tag{4}$$

If the rank of $\mathbf{A}$ is exactly 8, then the solution is unique (up to scale) and is given by the generator of the right-null space of $\mathbf{A}$. However, the data is often not exact due to noise in the point coordinates. The matrix $\mathbf{A}$ will then have a rank of 9. In this case, the solution is given by a least-squares solution. The least-squares solution for $\mathbf{f}$ is the vector that minimizes $||\mathbf{Af}||$ subject to $||\mathbf{f}|| = 1$. It can be proven that the solution to this optimization problem is the singular vector corresponding to the smallest singular value of $\mathbf{A}$. That is, the last column of $\mathbf{V}$ in the singular value decomposition (SVD) $\mathbf{A} = \mathbf{UDV}^T$.

Geometrically, $\mathbf{F}$ represents a mapping from the 2-dimensional projective plane of the first image to epipolar lines in the second image. Thus, the fundamental matrix represents a mapping from a 2-dimensional onto a 1-dimensional projective space, and must therefore have rank 2.

To enforce that $\mathbf{F}$ is of rank 2, $\mathbf{F}$ is replaced by a matrix $\mathbf{F}'$ that solves the following optimization problem

$$\min_{\mathbf{F}'} \quad ||\mathbf{F} - \mathbf{F}'|| \quad \text{s.t} \quad \text{rank}(\mathbf{F}')=2 \tag{5}$$

The solution is achieved by SVD. Let $\mathbf{F} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ where

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} \quad \text{and} \quad \mathbf{\Sigma}' = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{6}$$

The solution is then given by $\mathbf{F} = \mathbf{U}\mathbf{\Sigma}'\mathbf{V}^T$. This two-step algorithm of first estimating the fundamental matrix and then enforce its rank constraint is called the 8-point algorithm.

The measurement of data points is usually affected by noise. Further, eq. (2) is poorly conditioned where some elements will be much more sensitive to noise than others. Normalizing the data points before constructing the constraint matrix $\mathbf{A}$ will therefore yield a huge improvement in the conditioning of the problem and hence stability in the result. The suggested normalization is a translation and scaling of each image so that the centroid of the reference points is at the origin of the coordinates and the RMS distance of the points from the origin is equal to $\sqrt{2}$. When the data points are first normalized, the procedure of estimating the fundamental matrix is called the normalized 8-point algorithm.

Once the fundamental matrix $\mathbf{F}$ has been estimated, the epipolar line for any point $\mathbf{x}$ in the first image is given by $\mathbf{l}' = \mathbf{F}\mathbf{x}$. Similarly, $\mathbf{l} = \mathbf{F}^T\mathbf{x}'$ represents the epipolar line corresponding to $\mathbf{x}'$ in the second image.

3 Results

When extracting point matches, there are two things that should be considered in order to obtain a representative set of point matches. First, image points should be chosen spatially uniform over the whole image, and, secondly, the more image points, the better. Taking this into consideration, the chosen image points are illustrated in fig. 1.

Figure 1: Point correspondences.

The epipolar lines obtained from using the fundamental matrix estimated by the 8-point algorithm are shown in fig. 2.

Figure 2: Epipolar lines achieved from 8-point algorithm.

By using the normalized 8-point algorithm instead, the epipolar lines shown in fig. 3 were obtained.

Figure 3: Epipolar lines achieved from normalized 8-point algorithm.

Looking at the images in fig. 1, it can be argued that the motion between the views is a rotation and a slight translation. Hence, the images are taken with converging cameras. A useful property of converging cameras is that, in each image, the direction of the other camera can be deduced from the intersection of the epipolar lines. By comparing the epipolar geometry for converging cameras in fig. 4 with the views in fig. 1, it can be concluded that the epipolar lines for the normalized 8-point algorithm in fig. 3 coincide best with the geometry. Hence, the normalized algorithm yields a more valid result compared to the non-normalized algorithm. The epipolar lines in fig. 2 should have intersected on the other side of the image to be valid.

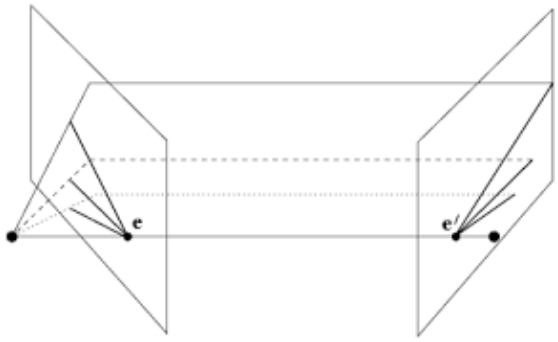

Figure 4: Epipolar geometry for converging cameras

4 Code

The main code for the fundamental matrix estimation is shown in listing 1. The code uses five helper functions that constructs the constraint matrix **A**, normalizes data, estimates the fundamental matrix **F** and visualizes the point correspondences and epipolar lines, respectively. The helper functions are shown in listing 2, listing 3, listing 4, listing 5 and listing 6.

Listing 1: Main code for fundamental matrix estimation.

```
clear variables;
%% Corresponding image points
img1 = [
    476     785     1;
    424     895     1;
    715     912     1;
    663     1014    1;
    32      1351    1;
    340     2204    1;
    278     2330    1;
    701     1950    1;
    1785    2617    1;
    1632    1958    1;
    2502    1837    1;
    1702    950     1;
    2449    650     1;
    1316    3245    1;
    1260    2048    1;
    1942    2352    1;
    771     153     1;
    2788    1349    1;
    151     210     1;
    1420    274     1;
    ]';

img2 = [
    696     1974    1;
    709     2082    1;
    934     1942    1;
    946     2046    1;
    624     2651    1;
    1334    3211    1;
    1348    3356    1;
    1457    2790    1;
    2788    2770    1;
    2209    2272    1;
    2886    1662    1;
    1702    1429    1;
    2132    773     1;
    2772    3646    1;
    1958    2562    1;
    2758    2437    1;
    621     1354    1;
    2822    1098    1;
    204     1720    1;
```

```
    1144    1089    1;
    ]';


%% Eight-Point Algorithm
A = constraint_matrix(img1,img2);
F = fundamental_matrix(A);


%% Normalized Eight-Point Agortihm
[img1_norm,T1] = data_normalization(img1);
[img2_norm,T2] = data_normalization(img2);

A_norm = constraint_matrix(img1_norm, img2_norm);
F_norm = fundamental_matrix(A_norm);
F_og = T2'*F_norm*T1;


%% Visualize results in images
% Draw circles at point correspondences
I1 = imread('hw05_f_img1.jpg');
I1_pts = draw_points(I1,img1);
I2 = imread('hw05_f_img2.jpg');
I2_pts = draw_points(I2,img2);

imwrite(I1_pts,'hw05_f_img1_pts.jpg');
imwrite(I2_pts,'hw05_f_img2_pts.jpg');

% Draw epipolar lines in img2
epi_lines_img2 = epipolarLine(F,img1(1:2,:)')';
I2_lines = draw_epipolar_lines(I2_pts,epi_lines_img2);
imwrite(I2_lines,'hw05_f_img2_epilines.jpg');

epi_lines_img2_norm = epipolarLine(F_og,img1(1:2,:)')';
I2_lines_norm = draw_epipolar_lines(I2_pts,epi_lines_img2_norm);
imwrite(I2_lines_norm,'hw05_f_img2_norm_epilines.jpg');

% Draw epipolar lines in img1
epi_lines_img1 = epipolarLine(F',img2(1:2,:)')';
I1_lines = draw_epipolar_lines(I1_pts,epi_lines_img1);
imwrite(I1_lines,'hw05_f_img1_epilines.jpg');

epi_lines_img1_norm = epipolarLine(F_og',img2(1:2,:)')';
I1_lines_norm = draw_epipolar_lines(I1_pts,epi_lines_img1_norm);
imwrite(I1_lines_norm,'hw05_f_img1_norm_epilines.jpg');
```

Listing 2: Code for normalizing data.

```
function [norm_data, T_data] = data_normalization(data) % ...
    Isotropic scaling
    [dim,npts] = size(data);
    if (dim == 3)   % 2D points given as homogeneous coordinates
        mean_data = mean(data,2); % Mean of each row (coordinate)
        dist_sum = 0;
        for i = 1:npts
            dist_sum = dist_sum+norm(data(:,i)-mean_data)^2;
```

```
        end
        scale_2 = sqrt(dist_sum/(2*npts));
        T_data = [
            1/scale_2, 0, -1/scale_2*mean_data(1);
            0, 1/scale_2, -1/scale_2*mean_data(2);
            0, 0, 1
            ];
        norm_data = T_data*data;
    else
       disp('Have not implemented data normalization for this ...
           point dimension');
    end
end
```

Listing 3: Code for constructing the constraint matrix **A**.

```
function A = constraint_matrix(img1,img2)
    % Asuming data (image points) are given as [x; y; w].
    x1 = img1(1,:)';
    y1 = img1(2,:)';
    x2 = img2(1,:)';
    y2 = img2(2,:)';
    [~,npts] = size(img1);
    A = [x2.*x1, x2.*y1, x2, y2.*x1, y2.*y1, y2, x1, y1, ...
        ones(npts,1)];
end
```

Listing 4: Code for estimating the fundamental matrix **F**.

```
function F = fundamental_matrix(A)
    % Extract fundamental matrix from the column of V ...
        corresponding to the
    % smallest singular value.
    [~,~,V] = svd(A);
    F = reshape(V(:,9),3,3)';

    % Enforce rank2 constraint
    [U,D,V] = svd(F);
    F = U*diag([D(1,1) D(2,2) 0])*V';
end
```

Listing 5: Code for visualizing point correspondences on image.

```
function RGB = draw_points(I,img_pts)
    [~,npts] = size(img_pts);
    radius = 20;
    RGB = I;
    for i = 1:npts
        x = img_pts(1,i);
        y = img_pts(2,i);
        RGB = ...
            insertShape(RGB,'FilledCircle',[x,y,radius],'Color','red');
```

```
    end
end
```

Listing 6: Code for visualizing epipolar lines on image.

```
function RGB = draw_epipolar_lines(I,lines)
    % Tried to use toBorderLinePoints instead of this function, but
    % something caused weird points to be calculated.
    [~,nlines] = size(lines);
    RGB = I;
    [x_size,~,~] = size(I);
    for i = 1:nlines
        a = lines(1,i);
        b = lines(2,i);
        c = lines(3,i);

        x1 = 0;
        x2 = x_size;
        y1 = -(a*x1+c)/b;
        y2 = -(a*x2+c)/b;

        RGB = ...
            insertShape(RGB,'Line',[x1,y1,x2,y2],'Color','blue','Linewidth',4);
    end
end
```

References

[1] *Fundamental matrix (computer vision)*. URL: `https://en.wikipedia.org/wiki/Fundamental_matrix_(computer_vision)` (visited on 05/11/2020).