VISUAL PERCEPTION

MULTIPLE VIEW GEOMETRY LAB REPORT

Gobichettipalayam Shriarulmozhivarman
Masters in Computer Vision

Submitted to Devesh Adlakha
Université de Bourgogne

20 May 2021

Chapter 1

Lab 2 Camera Calibration With Zhang Method

1.1 Camera Calibration

Camera calibration is the process of extracting the parameters of a camera,intrinsic parameters and extrinsic parameters from 2D image coordinates and 3D world coordinates .Intrinsic parameter are the camera's internal properties such as, its focal length, skew angle,and image centre.Extrinsic parameters of the camera are the 3D position and orientation in the world.In total we have 11 parameter to describe the pin hole camera Model that can be used to form images from the real world.Let m be the homogeneous coordinates of the image points and M be the Corresponding points in the World coordinates then,$m_i \cong PM_i$ is in equality up to a scale for all the index of the image,where P is the Projective matrix.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = P_{3,4} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1.1}$$

where $P = KR[I|-T]$,K is the intrinsic parameter,R is the rotation matrix and T is the position of the camera in the 3D world.As P is the unknown that we to find,let is be filled with variable names. And P be $P = \begin{bmatrix} P11 & P12 & P13 & P14 \\ P21 & P22 & P23 & P24 \\ P31 & P23 & P33 & P34 \end{bmatrix}$ so,

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} P11 & P12 & P13 & P14 \\ P21 & P22 & P23 & P24 \\ P31 & P32 & P33 & P34 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1.2}$$

Since we are dealing with a plane to plane homography the depth Z in the world coordinate s Zero(0).

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} P11 & P12 & P14 \\ P21 & P22 & P24 \\ P31 & P32 & P34 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{1.3}$$

this of the form $m \cong K[r_1, r_2 t]M$ where K is the internal parameter, and $r_1, r_2 and t$ are the rotational and camera location.

$$x = \frac{p_{11}X + p_{12}Y + p_{14}}{p_{31}X + p_{32}Y + p_{34}} \tag{1.4}$$

$$y = \frac{p_{21}X + p_{22}Y + p_{24}}{p_{31}X + p_{32}Y + p_{34}} \tag{1.5}$$

we have 2 equations for a point and 12 unknowns, so we at-least need 6 points to get all the parameters. this is can be done by two methods equation rearranging and Kronecker product methods.For the linear equation to get the parameters we have to rearrange the equation in such a way the we solve for $AX = 0$.

Let us define $A = \begin{bmatrix} P_{11} \\ P_{12} \\ P_{14} \end{bmatrix}, B = \begin{bmatrix} P_{21} \\ P_{22} \\ P_{24} \end{bmatrix}, B = \begin{bmatrix} P_{31} \\ P_{32} \\ P_{34} \end{bmatrix}$ which implies that $m \cong PM$ will be

$$\begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = \begin{bmatrix} A^T \\ B^T \\ C^T \end{bmatrix} Mi \tag{1.6}$$

and $x_i = A^T Mi$, $y_i = B^T Mi$,$w_i = C^T Mi$,so that $x_i = \frac{A^T Mi}{C^T Mi}$ and $y_i = \frac{B^T Mi}{C^T Mi}$ and rearrange it as,

$$-M_i^T A + 0 + x_i M_i^T C = 0$$
$$0 + -M_i^T B + x_i M_i^T C = 0$$

$$\alpha_{xi}^T p = 0$$
$$\alpha_{yi}^T p = 0$$

where α_{xi}^T is $[-M_i^T, 0^T, x_i M_i^T]$, α_{yi}^T is $[0^T, -M_i^T, y_i M_i^T]$ and p = $\begin{bmatrix} A \\ B \\ C \end{bmatrix}_{12,1}$

stacking the α_{xi}^T and α_{yi}^Tif we stack n points to have 2n x12 matrix,this can be solved by the Singular value decomposition(SVD), as this is of form $AX = 0$.stacking atleast 6 points,ensures a solution that can be used to extract the parameters from the projection Matrix(Homography).

$K\begin{bmatrix} h1 & h2 & h3 \end{bmatrix} = K\begin{bmatrix} r1 & r2 & t \end{bmatrix}$, so we can use the constrains, Using the knowledge that r1 and r2 are orthonormal, then

$$h_1^T K^{-T} K^{-1} h_2 = 0 \tag{1.7}$$

and

$$h_1^T K^{-T} K^{-1} h_1 = h_2^T K^{-T} K^{-1} h_2 \tag{1.8}$$

$$B = K^{-T} K^{-1} \tag{1.9}$$

B is symmetric that can be defined by a 6D vector b =$[B_{11} B_{12} B_{22} B_{13} B_{23} B_{33}]$, with i^{th} column of the H be h_i, then $h_i^T B h_j = V_{ij}^T b$.

$$\begin{bmatrix} v_{12}^T \\ (v_{11} - v_{22})^T \end{bmatrix} b = 0 \tag{1.10}$$

where v_{ij} is

$$v_{i,j} = \begin{bmatrix} h_{i1} h_{j1} \\ h_{i1} h_{j2} + h_{i2} h_{j1} \\ h_{i2} h_{j2} \\ h_{i3} h_{j1} + h_{i1} h_{j3} \\ h_{i3} h_{j2} + h_{i2} h_{j3} \\ h_{i3} h_{j3} \end{bmatrix}$$

This is implemented in the getV.m file,then it can be solved SVD for $Vb = 0$, where V is the 2n x 6 matrix.Once we have the then we compute the K intrinsic parameters by:

$$vo = (B_{12} B_{13} - B_{11} B_{23}) / (B_{11} B_{22} - B_{12}^2)$$

$$\lambda = B_{33} - [B_{13}^2 + vo(B_{12} B_{13} - B_{11} B_{23})] / B_{11}$$

$$\alpha = \sqrt{\lambda / B_{11}}$$

$$\beta = \sqrt{\lambda B_{11} / (B_{11} B_{22} - B_{12}^2)}$$

$$\gamma = -B_{12} \alpha^2 \beta / \lambda$$

$$uo = \gamma vo / \beta - B_{13} \alpha^2 / \lambda$$

Having obtained the values of the we can rearrange to get the K matrix

The main code implementation is done in the lab2.m.

```
>> load('K.txt')
>> K

K =

   300     0   128
     0   300   128
     0     0     1
```

(a) Actual Intrinsic parameters

```
K =

  300.0000    0.0000  128.0000
         0  300.0000  128.0000
         0         0    1.0000
```

(b) Estimated Intrinsic parameters from matlab implementation

Figure 1.1: Comparison on the Actual and Estimated Intrinsic parameters

The Zhang calibration with the noise is implemented in the lab2_withnoise.m

```
K =

  295.5191   -0.3161  123.0212
         0  295.0237  128.8133
         0         0    1.0000
```

(a) Intrinsic parameters for 0.5 std Gaussian error

```
K =

  291.5247   -1.4972  111.7565
         0  290.8193  125.5412
         0         0    1.0000
```

(b) Intrinsic parameters for 1 std Gaussian error

Figure 1.2: Comparison on the Errors with 0.5 and 1 Std of the Gaussian Error.

With the increase in the standard deviation of the Gaussian error the Intrinsic parameters estimated are moving away from the actual parametes

Chapter 2

Lab 4 Epipolar geometry: Fundamental Matrix

The fundamental matrix is mathematical expression of scenes taken with two uncalibrated cameras.In homogeneous image coordinates x and x', of corresponding points in a stereo image pair, Fx is the epipolar line on which the corresponding point x' on the other image must be on.This gives the constrain that is x'Fx=0 where the F is the fundamental matrix.One point gives one equation,and the fundamental matrix is rank 2 and with DOF of 8, so we shall need atleast 8 points for the estimation the fundamental matrix.And we donot need any information of the calibration parameters for the estimation of the fundamental matrix.

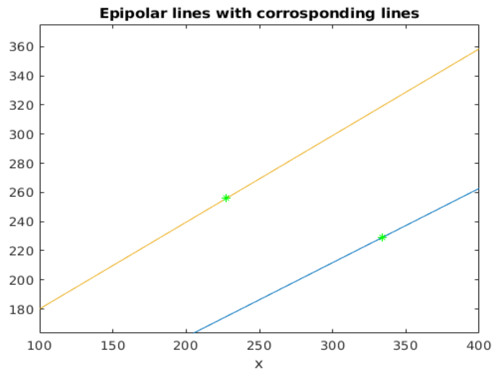
The Epipolar lines interest at the epipole this can be computed by the computing the epipolar lines and cross product of the gives the epipole.

$$\begin{bmatrix} x_1 & y_1 & 1 \end{bmatrix} \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{23} & F_{33} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = 0 \tag{2.1}$$

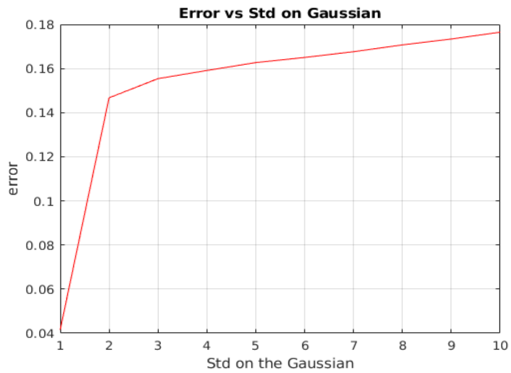
Rearrangeing the equation above in the from Ax = 0, where x is the elements of the fundamental matrix.This can be solved by the SVD.

Matlab Implementation

I have implemented the main function in the file lab4.m which calls the ComputeFundamentalMatrix.m to get the fundamental matrix when first image points and its corresponding points in a stereo image pair are the inputed in the homogeneous from.



(a) EpipolarLines on the point



(b) absolute Error with the rising std

Figure 2.1: Epipolar Geometry

From the Results in the lab the error is high without the Hartley's Pre-processing.The error increases with the increase of the standard deviation on Gaussian Noised added to the images.

Matlab Code

Listing 2.1: lab2.m

```
clc
clear all

% Loading the world coordinates
checkerboardpoints = load('ptsXY.txt');
checkerboardpoints(:,3) = 1;

L = [];

% estimating the homography for each image
for j = 1:10
    fileName = append('pts2D_',int2str(j),'.txt');
    imagePoints = load(fileName);
    imagePoints(3,:) = 1;
    A = [];
    for i = 1:100
%          stacking the point equation
        alphax = [-checkerboardpoints(i,:),0,0,0,(imagePoints(1,i).*checkerboardpoints(i,:))];
        A = vertcat(A,alphax);
        alphay = [0,0,0,-checkerboardpoints(i,:),(imagePoints(2,i).*checkerboardpoints(i,:))];
        A = vertcat(A,alphay);
    end
    % estamating the Homography
    [U,S,V] = svd(A);
```

```

    J = V(:,end);
    A = J(1:3)';
    B = J(4:6)';
    C = J(7:9)';
    H = [A;B;C];


    % stacking the v vector
    L= vertcat(L ,[getV(H,1,2)';
     (getV(H,1,1)-getV(H,2,2))']);

end

%computing the B matrix
[U,S,V] = svd(L);
B =V(:,end);

% computing the parameters
vo = (B(2)*B(4)-B(1)*B(5))/(B(1)*B(3)-B(2)^2) ;
lambda = B(6)- ((B(4)^2+vo*(B(2)*B(4)-B(1)*B(5)))/B(1));
alpha = sqrt(lambda/B(1));
beta = sqrt((lambda*B(1))/(B(1)*B(3)-B(2)^2));
gamma = -B(2)*alpha^2*beta/lambda;
uo = (gamma*vo/beta)-(B(4)*alpha^2/lambda);


% rearraing to get the K matrix
K = [alpha gamma uo;
    0   beta vo;
    0    0      1]
```

Listing 2.2: lab2_withnoise.m

```
clc
clear all
```

```

% Loading the world coordinates
checkerboardpoints = load('ptsXY.txt');
checkerboardpoints(:,3) = 1;

% estimating the homography for each image
L = [];
% computing the gaussian error
gaussian = fspecial('gaussian',size(checkerboardpoints),1);
for j = 1:10
    fileName = append('pts2D_',int2str(j),'.txt');
    imagePoints = load(fileName);
    imagePoints(3,:) = 1;
    imagePoints = gaussian'+imagePoints;
    imagePoints = imagePoints./imagePoints(3,:);
    A = [];
    for i = 1:100
        alphax = [-checkerboardpoints(i,:),0,0,0,(imagePoints
           (1,i).*checkerboardpoints(i,:))];
        A = vertcat(A,alphax);
        alphay = [0,0,0,-checkerboardpoints(i,:),(imagePoints
           (2,i).*checkerboardpoints(i,:))];
        A = vertcat(A,alphay);
    end

    [U,S,V] = svd(A);

    J = V(:,end);
    A = J(1:3)';
    B = J(4:6)';
    C = J(7:9)';
    H = [A;B;C];

    % stacking the v vector
    L= vertcat(L ,[getV(H,1,2)';
```

```
     (getV(H,1,1)-getV(H,2,2))']);

end

%computing the B matrix
[U,S,V] = svd(L);
B =V(:,end);


% computing the parameters
vo = (B(2)*B(4)-B(1)*B(5))/(B(1)*B(3)-B(2)^2)  ;
lambda = B(6)- ((B(4)^2+vo*(B(2)*B(4)-B(1)*B(5)))/B(1));
alpha = sqrt(lambda/B(1));
beta = sqrt((lambda*B(1))/(B(1)*B(3)-B(2)^2));
gamma = -B(2)*alpha^2*beta/lambda;
uo = (gamma*vo/beta)-(B(4)*alpha^2/lambda);

% rearraing to get the K matrix
K = [alpha gamma uo;
    0   beta  vo;
    0    0     1]
```

Listing 2.3: getV.m Helper function for lab2

```
function V = getV(H,i,j)
%% getV
%          Rearranges the Homography matrix
%    Input
%        H - Homography matrix
%        i - Index
%        j - index
%
%    Output
%        V - Rearranged form of H
%% Function starts here
    V =[H(1,i)*H(1,j);
```

```
        H(1,i)*H(2,j)+H(2,i)*H(1,j);
        H(2,i)*H(2,j);
        H(3,i)*H(1,j)+H(1,i)*H(3,j);
        H(3,i)*H(2,j)+H(2,i)*H(3,j);
        H(3,i)*H(3,j)];
end
```

Listing 2.4: lab4.m lab-4 main script

```

clc
clear all

%loading the image points
imageP1 = load('pts2D_1.txt');
imageP2 = load('pts2D_2.txt');


% converting image coordincates to homogenoues form
imageP1(:,3) = 1;
imageP2(:,3) = 1;

%computing the FundamantalMatrix
F_matrix = ComputeFundamentalMatrix(imageP1,imageP2);

%computing the error
error = [];
for i = 1:300
    Z = imageP1(i,:)*F_matrix*imageP2(i,:)';
    error = [error;Z];
end
% summing all the error
Error = sum(abs(error));

%selecting random points
i = randi(300);
```

```
j = randi(300);

%ploting the epipolar line and the image point
polar_line1 = F_matrix*imageP2(i,:)';
polar_line1(:,:) = polar_line1(:,:)./-polar_line1(2,:);
f = @(x) polar_line1(1,:)*x+polar_line1(3,:);
ezplot( f, 100, 400)
hold on
plot(imageP1(i,1),imageP1(i,2),'g*')
title('Epipolar lines on the image point')

% computing the epipoles
polar_line1 = F_matrix*imageP1(i,:)';
polar_line2 = F_matrix*imageP1(j,:)';

epipoles = cross(polar_line1',polar_line2')

polar_line1 = F_matrix*imageP2(i,:)';
polar_line2 = F_matrix*imageP2(j,:)';

epipoles = cross(polar_line1',polar_line2')

%%
clear all
close all


errors = [];
for i = 1:10
%loading the image points

    imageP1 = load('pts2D_1.txt');
    imageP2 = load('pts2D_2.txt');

    % converting image coordincates to homogenoues form
```

```
    imageP1(:,3) = 1;
    imageP2(:,3) = 1;
%   Generating the Gaussian error and adding it to the image
    gaussian = fspecial('gaussian',size(imageP1),0.5*i);
    imageP1 = imageP1+gaussian;
    imageP2 = imageP2+gaussian;


    imageP1 = imageP1./imageP1(:,3);
    imageP2= imageP2./imageP2(:,3);

%computing the FundamantalMatrix on the noised images

    F_matrix = ComputeFundamentalMatrix(imageP1,imageP2);
% calculating the error
    error = [];
    for i = 1:300
        Z = imageP1(i,:)*F_matrix*imageP2(i,:)';
        error = [error;Z];
    end
    sumError = sum(abs(error));
    errors = [errors;sumError];
end
%ploting the error vs std
plot(1:10,errors,'-r')
grid on
title('Error vs Std on Gaussian')
xlabel('Std on the Gaussian')
ylabel('error')
```

Listing 2.5: ComputeFundamentalMatrix.m Helper function for lab-4

```
function F_matrix = ComputeFundamentalMatrix(imageP1,imageP2)
%% ComputeFundamentalMatrix
        %form the image coordinates with its corrosponding
           image poins in
```

```
%         the second image compute the fundamental matrix
%   Input
%       imageP1 - Image coordinate points of a image in
   homogeoeous form
%       imageP2 - corrosponding image poin in the streo in
   homogeoeous form
%
%   Output
%       F_matrix - Fundamental Matrix of the streo images
%% Function starts here

% Preprocessing
% the centroid of the transformed points is at the origin and
% the average distance of the transformed points to the
   origin is sqrt(2)
% from the Hartley, Richard I. "In defense of the eight-point
    algorithm."


% processing on the imageP1
    mean_p1 = mean(imageP1);
    Centred_p1 = imageP1 - repmat(mean_p1, [size(imageP1,1),
       1]);
    var_p1 = var(Centred_p1);
    sd_p1 = sqrt(var_p1);
    Tp1 = [1/sd_p1(1), 0,0; 0,1/sd_p1(2), 0; 0,0,1]*[1,0,-
       mean_p1(1);0,1,-mean_p1(2);0,0,1];
    Normalized_p1 = Tp1 * [imageP1'];

% processing on the imageP2
    mean_p2 = mean(imageP2);
    Centred_p2 = imageP2 - repmat(mean_p2, [size(imageP2,1),
       1]);
    var_p2 = var(Centred_p2);
    sd_p2 = sqrt(var_p2);
```

```
    Tp2 = [1/sd_p2(1), 0,0; 0,1/sd_p2(2), 0; 0,0,1]*[1,0,-
       mean_p2(1);0,1,-mean_p2(2);0,0,1];
    Normalized_p2 = Tp2 * [imageP2]';

    % stacking the equation
    J = [];
    for i = 1:size(imageP1,1)
        P1 = Normalized_p1(:,i);
        P2 = Normalized_p2(:,i);
        K = [P1(1)*P2(1) P1(2)*P2(1) P2(1) P2(2)*P1(1) P2(2)*
           P1(2) P2(2) P1(1) P1(2) 1];
        J = [J; K];
    end

    %solving for the fundamental matrix

    [U,S,V] = svd(J);

    F = V(:,end);
    F_rank3 = reshape(F,[3,3]);

    % enforcing the rank 2 condition
    [U,S,V] = svd(F_rank3);
    S(3,3) = 0 ;
    F_rank2 = U*S*V';

    % undoing the preprocessing process to get the
       fundamental matrix
    F_matrix = Tp1' * F_rank2 * Tp2;
end
```