Homework 4 Writeup

Instructions

- Describe any interesting decisions you made to write your algorithm.
- Show and discuss the results of your algorithm.
- Feel free to include code snippets, images, and equations.
- Use as many pages as you need, but err on the short side If you feel you only need to write a short amount to meet the brief, th
- **Please make this document anonymous.**

Files&Algorithm

1. bayer_to_rgb_bicubic.m

input : bayer_img (RGBG)
output : rgb_img (RGB)

I used bilinear interpolation to change bayer image to rgb image.
1) Make zero image.
2) Get size of bayer image.
3) There are two case of Green. First, $mod(i,2) == 0 \&\& mod(j,2) == 1$.
Second is $mod(i,2) == 1 \&\& mod(j,2) == 0$.
4) Case of Red is mod(i,2) == 1. Else is case of Blue

2. calculate_fundamental_matrix.m

input : pts1, pts2 (feature point)
output : f (Fundamental matrix)

1) Make array A.
2) Using svd function, get smallest eigenvector of $A^T A$.
3) $F = reshape(eigenvector)$.
4) $F = USV^T$
5) Make minimum singular value for S become zero.
6) $F = USV^T$ by using modified S at 5)

3. calculate_rectification_matrix.m

input : f (Fundamental matrix), imageSize, pts1, pts2
output : t1, t2

1) Make u,d,v of f with svd function.
2) Make $epipole = u_3$
3) $t = [1\ 0\ -w./2;\ 0\ 1\ -h./2;\ 0\ 0\ 1]$
4) $theta = atan(p2t(2)/p2t(1));$
5) $r = [cos(-theta)\ -sin(-theta)\ 0;\ sin(-theta)\ cos(-theta)\ 0;\ 0\ 0\ 1]$
6) $g = [1\ 0\ 0;\ 0\ 1\ 0;\ -1/ex\ 0\ 1]$

4. calculat_disparity_map.m

Make a cost volume using NCC(normalized cross correlation) matching cost function for the two rectified images, then obtain disparity map from the cost volume after aggregate it with a box filter.

In code.

1. bayer_to_rgb_bicubic.m

```
function rgb_img = bayer_to_rgb_bicubic(bayer_img)
    [M,N,L] = size(bayer_img);
    img = uint8(zeros(M,N,3));
    for i = 2:M-1
        for j = 2:N-1
            if mod(i,2) == 0 && mod(j,2) == 1 \%G
                img(i,j,1)=round((bayer_img(i-1,j)+
                   bayer_img(i+1,j))/2);
                img(i,j,2)=round(bayer_img(i,j));
                img(i,j,3)=round((bayer_img(i,j-1)+
                   bayer_img(i,j+1))/2);
            elseif mod(i,2) == 1 && mod(j,2) == 0
                img(i,j,1)=round((bayer_img(i,j-1)+
                   bayer_img(i,j+1))/2);
                img(i,j,2)=round(bayer_img(i,j));
                img(i,j,3)=round((bayer_img(i-1,j)+
                   bayer_img(i+1,j))/2);
            elseif mod(i,2) == 1 \%R
                img(i,j,1)=round(bayer_img(i,j));
```

```
                img(i,j,2)=round((bayer_img(i-1,j)+
                   bayer_img(i+1,j)+bayer_img(i,j-1)+
                   bayer_img(i,j+1))/4);
                img(i,j,3)=round((bayer_img(i-1,j-1)+
                   bayer_img(i+1,j-1)+bayer_img(i+1,j-1)+
                   bayer_img(i-1,j+1))/4);
            else \%B
                img(i,j,1)=round((bayer_img(i-1,j-1)+
                   bayer_img(i+1,j-1)+bayer_img(i+1,j-1)+
                   bayer_img(i-1,j+1))/4);
                img(i,j,2)=round((bayer_img(i-1,j)+
                   bayer_img(i+1,j)+bayer_img(i,j-1)+
                   bayer_img(i,j+1))/4);
                img(i,j,3)=round(bayer_img(i,j));
            end
        end
    end
    bayer_img = img;
    rgb_img = bayer_img;
```

2. calculate_fundamental_matrix.m

```
function f = calculate_fundamental_matrix(pts1, pts2)
    [m,~] = size(pts1);
    X1 = pts1(:, 1);    Y1 = pts1(:, 2);
    X2 = pts2(:, 1);    Y2 = pts2(:, 2);
    A = [X1.*X2, X1.*Y2, X1, Y1.*X2, Y1.*Y2, Y1, X2, Y2,
       ones(m,1)];

    [~,~,EV1] = svd(A);

    EV = EV1(:,9);

    F = reshape(EV,3,3);

    [U,S,V] = svd(F);
    S(3,3) = 0;
    F = U*S*V';
    f = F;
```

$$A = \begin{bmatrix} xx' & xy' & x & yx' & yy' & y & x' & y' & 1 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ xx' & xy' & x & yx' & yy' & y & x' & y' & 1 \end{bmatrix}$$

3. gen_hybrid_image.m

```
function [t1, t2] = calculate_rectification_matrix(f,
   imageSize, pts1, pts2)
    [u, d, v] = svd(f);
    epipole = u(:, 3);
    ep = epipole/epipole(3);
    h = imageSize(1);
    w = imageSize(2);
    t = [1 0 -w./2; 0 1 -h./2; 0 0 1];
    p2t = t*ep;
    theta = atan(p2t(2)/p2t(1));
    r = [cos(-theta) -sin(-theta) 0; sin(-theta) cos(-
       theta) 0; 0 0 1];
    p2r = r*p2t;
    if (abs(p2r(3)/norm(p2r))< 1e-6)
        g = eye(3);
    else
        ex = p2r(1)/p2r(3);
        g = [1 0 0; 0 1 0; -1/ex 0 1];
    end
```

Calculate g, r, t with algorithm in Files & Algorithm section.

4. calculat_disparity_map.m

```
function d = calculate_disparity_map(img_left, img_right,
    window_size, max_disparity)
    img_left = im2double(img_left);
    img_right = im2double(img_right);
    [h w] = size(img_left);
    cost_vol = zeros(h,w,max_disparity);

    f = [1,1,1; 1,1,1; 1,1,1];
    meanl = imfilter(img_left,f/9);
    meanr = imfilter(img_right,f/9);

    difl = img_left - meanl;
    difr = img_right - meanr;

    sql = difl.*difl;
    sqr = difr.*difr;
```

```
16
17      suml = sqrt(imfilter(sql, f));
18      sumr = sqrt(imfilter(sqr, f));
19
20      for i = 1:max_disparity
21          l = difl(:,1:end-i);
22          r = difr(:, i+1:end);
23          lr = l.*r;
24          sum_lr = imfilter(lr,f);
25
26          l2 = suml(:,1:end-i);
27          r2 = sumr(:,i+1:end);
28
29          cost_vol(:,1:end-i,i) = sum_lr./(l2.*r2);
30      end
```

Result...

1. bayer_to_rgb_bicubic.m

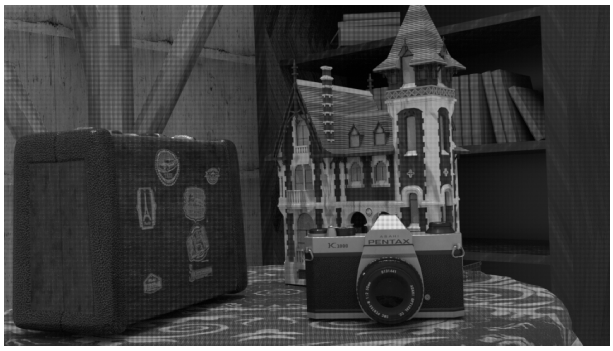

Figure 1: bayer image.

Figure 2: bayer to rgb image.

2. calculate_fundamental_matrix.m

3. calculate_rectification_matrix.m

Figure 3: visualize matching points on images.

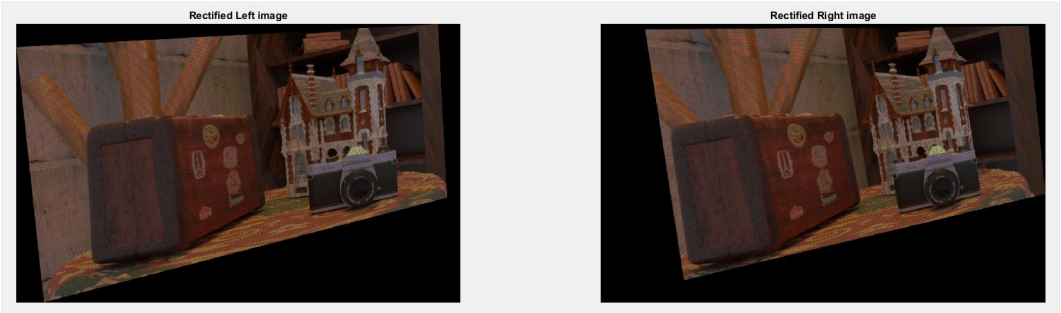

Figure 4: visualize rectified images.

4. calculat_disparity_map.m

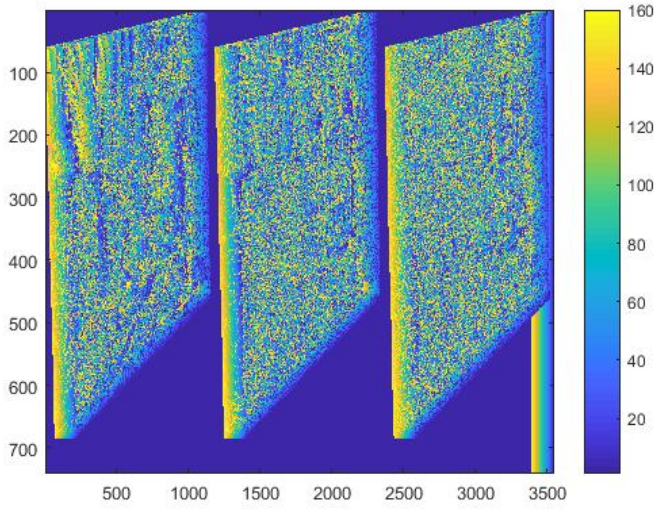

Figure 5: visualize rectified images.