

## Homework 5 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, then
- **Please make this document anonymous.**

### Interesting Implementation Detail

#### 1. My code about bilinear interpolation

```
1      [n, m, f] = size(bayer_img);
2
3      rgb_img = zeros(n,m,3);
4
5      for i = 1:n
6          for j = 1:m
7              if mod(i, 2) == 1 && mod(j, 2) == 1
8                  rgb_img(i, j, 1) = bayer_img(i, j);
9              elseif mod(i, 2) == 1 && mod(j, 2) == 0
10                 if j == m
11                     rgb_img(i, j, 1) = bayer_img(i, j - 1);
12                 else
13                     rgb_img(i, j, 1) = (bayer_img(i, j-1) +
14                                         bayer_img(i, j+1)) / 2;
15                 end
16             elseif mod(i, 2) == 0 && mod(j, 2) == 1
17                 if i == n
18                     rgb_img(i, j, 1) = bayer_img(i - 1, j);
19                 else
20                     rgb_img(i, j, 1) = (bayer_img(i - 1, j)
21                                         + bayer_img(i + 1, j)) / 2;
22                 end
23             elseif mod(i, 2) == 0 && mod(j, 2) == 0
24                 if i == n && j == m
25                     rgb_img(i, j, 1) = bayer_img(i-1, j-1);
26                 elseif i == n
```

```

25         rgb_img(i,j,1) = (bayer_img(i-1,j-1)
26             + bayer_img(i-1,j+1)) / 2;
27     elseif j == m
28         rgb_img(i,j,1) = (bayer_img(i-1,j-1)
29             + bayer_img(i+1,j-1)) / 2;
30     else
31         rgb_img(i,j,1) = (bayer_img(i-1,j-1)
32             + bayer_img(i-1,j+1) + bayer_img(i
33             +1,j-1) + bayer_img(i+1,j+1)) / 4;
34     end
35 end
36 end
37 end
38
39 for i = 1:n
40     for j = 1:m
41         if mod(i, 2) == 0 && mod(j, 2) == 0
42             rgb_img(i, j, 3) = bayer_img(i, j);
43         elseif mod(i, 2) == 0 && mod(j, 2) == 1
44             if j == 1
45                 rgb_img(i,j,3) = bayer_img(i,j + 1);
46             else
47                 rgb_img(i,j,3) = (bayer_img(i,j-1) +
48                     bayer_img(i,j+1)) / 2;
49             end
50         elseif mod(i, 2) == 1 && mod(j, 2) == 0
51             if i == 1
52                 rgb_img(i,j,3) = bayer_img(i + 1,j);
53             else
54                 rgb_img(i,j,3) = (bayer_img(i - 1,j)
55                     + bayer_img(i + 1,j)) / 2;
56             end
57         elseif mod(i, 2) == 1 && mod(j, 2) == 1
58             if i == 1 && j == 1
59                 rgb_img(i,j,3) = bayer_img(i+1,j+1);
60             elseif i == 1
61                 rgb_img(i,j,3) = (bayer_img(i+1,j-1)
62                     + bayer_img(i+1,j+1)) / 2;
63             elseif j == 1
64                 rgb_img(i,j,3) = (bayer_img(i-1,j+1)
65                     + bayer_img(i+1,j+1)) / 2;
66             else
67                 rgb_img(i,j,3) = (bayer_img(i-1,j-1)
68                     + bayer_img(i-1,j+1) + bayer_img(i
69                     +1,j-1) + bayer_img(i+1,j+1)) / 4;
70             end
71         end
72     end
73 end

```

```

61         end
62     end
63 end
64 end
65
66 for i = 1:n
67     for j = 1:m
68         if mod(i,2) ~= mod(j,2)
69             rgb_img(i,j,2) = bayer_img(i,j);
70         else
71             if i == 1 && j == 1
72                 rgb_img(i,j,2) = (bayer_img(i,j+1) +
73                                     bayer_img(i+1,j)) / 2;
74             elseif i == n && j == m
75                 rgb_img(i,j,2) = (bayer_img(i,j-1) +
76                                     bayer_img(i-1,j)) / 2;
77             elseif i == 1
78                 rgb_img(i,j,2) = (bayer_img(i,j-1) +
79                                     bayer_img(i,j+1) + bayer_img(i+1,j)
80                                     ) / 3;
81             elseif i == n
82                 rgb_img(i,j,2) = (bayer_img(i,j-1) +
83                                     bayer_img(i,j+1) + bayer_img(i-1,j)
84                                     ) / 3;
85             elseif j == 1
86                 rgb_img(i,j,2) = (bayer_img(i-1,j) +
87                                     bayer_img(i,j+1) + bayer_img(i+1,j)
88                                     ) / 3;
89             elseif j == m
90                 rgb_img(i,j,2) = (bayer_img(i-1,j) +
91                                     bayer_img(i,j-1) + bayer_img(i+1,j)
92                                     ) / 3;
93             else
94                 rgb_img(i,j,2) = (bayer_img(i-1,j) +
95                                     bayer_img(i+1,j) + bayer_img(i,j
96                                     -1) + bayer_img(i,j+1))/ 4;
97             end
98         end
99     end
100 end
101
102 rgb_img = cast(rgb_img, 'uint8');

```

When I fill missing R, G, B values using bayer image, I divide 4 case, (number of values that are interpolated is 1, 2, 3, 4) with individually R, G, B case. For example, filling corner of R, B value, I use 1 points, interpolating edge of R, B value and corner of G

value, I use 2 points. And interpolating edge of G value, I use 3 points, another cases, I use 4 points.

## Result

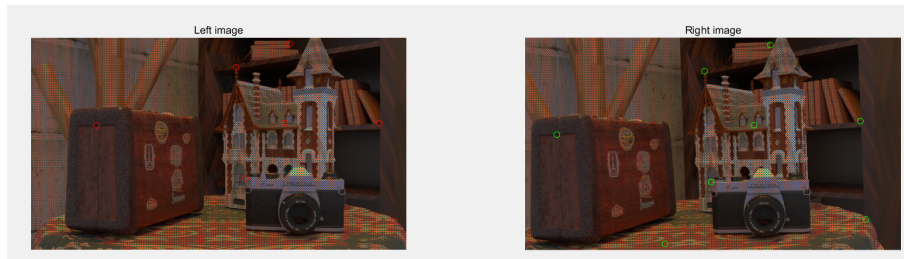


Figure 1: My result for bilinear interpolation.

## 2. Finding Fundamental matrix

```
1
2     [n, m] = size(pts1);
3
4     A = zeros(n, 9);
5
6     for i=1:n
7         x = pts1(i,1);
8         y = pts1(i,2);
9         x2 = pts2(i,1);
10        y2 = pts2(i,2);
11        A(i,1) = x*x2;
12        A(i,2) = x*y2;
13        A(i,3) = x;
14        A(i,4) = y*x2;
15        A(i,5) = y*y2;
16        A(i,6) = y;
17        A(i,7) = x2;
18        A(i,8) = y2;
19        A(i,9) = 1;
20    end
21
22    A2 = A.' * A;
23    [v, e] = eig(A2);
24
25    k = 1;
26    eigen_value = e(k,k);
27    f = v(:,k);
28    f = f / norm(f);
29    F = zeros(3,3);
```

```
30     for i=1:3
31         for j=1:3
32             F(i,j) = f(i + (j-1) * 3);
33         end
34     end
35
36     [U, S, V] = svd(F);
37     S(3,3) = 0;
38     F2 = U*S*V.';
39
40     f = F2;
```

I followed supplementary pdf's flow.

### 3. Finding Homography matrix for image rectification

```
1     w = imageSize(2);
2     h = imageSize(1);
3     t = eye(3);
4     t(1,3) = -w / 2;
5     t(2,3) = -h / 2;
6     e = epipole / epipole(3);
7     e2 = t * e;
8
9     ang = atan2(e2(2), e2(1));
10    co = cos(-ang + pi);
11    si = sin(-ang + pi);
12
13    r = eye(3);
14    r(1,1) = co;
15    r(1,2) = -si;
16    r(2,1) = si;
17    r(2,2) = co;
18
19    e3 = r * e2;
20    g = eye(3);
21    g(3,1) = -1 / e3(1);
```

I followed supplementary pdf's flow too. To make z's coordinate to 1, I execute normalizing first. I find Rotation matrix that make y coordinate to 1 and based on z-axis

Result



Figure 2: My result for image rectification

#### 4. Stereo matching and disparity map

```

1  s = size(img_right);
2  w = s(2);
3  h = s(1);
4
5  cost_vol = zeros(h, w, max_disparity);
6
7  pad_left = padarray(img_left, [0 max_disparity], -1,
8  'post');
9
10 block_pad = round(window_size/2) - 1;
11 padded_left = padarray(pad_left, [block_pad block_pad
12  ], 'replicate', 'both');
13
14 pad_right = padarray(img_right, [block_pad block_pad
15  ], 'replicate', 'both');
16
17 filter = zeros(window_size, window_size);
18
19 filter(1:window_size, 1:window_size) = 1 / (
20  window_size^2);
21
22 avg_right = imfilter(img_right, filter, 'replicate');
23 avg_left = imfilter(pad_left, filter, 'replicate');
24
25 a = zeros(window_size^2);
26 b = zeros(window_size^2);
27 for i=1:h
28     for j=1:w
29         for d = 1:max_disparity
30             avg_l = avg_left(i, j + d);
31             avg_r = avg_right(i, j);
32             A = padded_left(i:i+block_pad+1, j+d:j+d+

```

```

30         block_pad+1);
31         a = A(:) - avg_l;
32         B = pad_right(i:i+block_pad+1, j:j+
33             block_pad+1);
34         b = B(:) - avg_r;
35         cost_vol(i,j,d) = -dot(a,b) / (norm(a) *
36             norm(b));
37     end
38 end
39 end

```

To up executed speed, I used average filter to calculate average of all [window size \* window size] subarray first. These average values are used when get NCC score. NCC score's range is [-1, 1]. When NCC score is 1, I can say this is matched. So I multiple -1 to NCC score for change min score mean matching.

5. Aggregate the cost volume using box filter.

```

1     for d = 1:max_disparity
2         cost_vol(:, :, d) = imfilter(cost_vol(:, :, d),
3             filter, 'replicate');
4     end

```

I aggregate the cost volume individually d value with 2D box filter.

Result

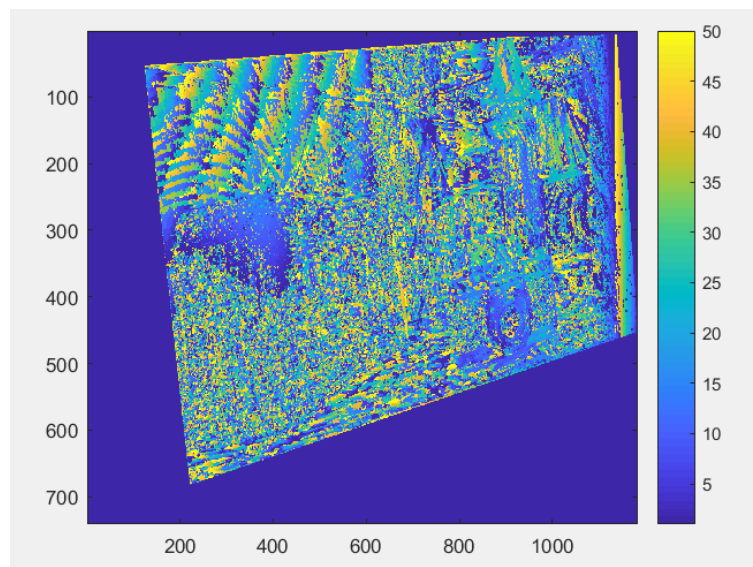


Figure 3: My result for finding disparity map. (window size = 3, max disparity = 50)