# **Computational Photography Assignment #5 by Junekyo Jhung**

# **Initials**

## Description

과제에서 기본으로 제공된 chessboard\_lightfield.png 이미지를 불러들인 후 5차원의 L(u; v; s; t; c) 배열을 정의한다. 변수 u와 v는 aperture, s 와 t는 lenslet, c 는 컬러 채널이다. 문제에서 제시한 것을 바탕으로 각각의 변수를 정의한다.

## MATLAB

%% Initials

disp('Loading the chess image..');

img = imread('data/chessboard\_lightfield.png');

[w, h, ch] = size(img); % width, height, channel

u = 16; % aperture coordinate u

v = 16; % aperture coordinate v

s = 400; % lenslet coordinate s

t = 700; % lenslet coordinate t

c = 3; % color channels

L = zeros(u, v, s, t, c); % 5-dimentional array of light field

# **Sub-aperture Views**

## Description

Camera aperture 변수인 u와 v를 변경(u0, v0)하여 각기 다른 pinhole에서의 이미지를 하나의 모자이크 이미지로 합성하는 것이 목적이다. L의 aperture domain에 pinhole image를 갱신하여 하나의 이미지를 합성하여 mosaic image를 figure 1과 같이 만들 수 있다.

## MATLAB

%% Sub-aperture Views

disp('Creating mosaic image of sub-aperture views..');

for i = 1:u

for j = 1:v

for k = 1:s

for l = 1:t

u\_o = u \* (k - 1) + i;

v\_o = v \* (l - 1) + j;

L(i, j, k, l, 1) = double(img(u\_o, v\_o, 1)) / 256;

L(i, j, k, l, 2) = double(img(u\_o, v\_o, 2)) / 256;

L(i, j, k, l, 3) = double(img(u\_o, v\_o, 3)) / 256;

end

end

end

end

for i = 1:u

for j = 1:v

img\_subapt((s \* (i - 1) + 1):(s \* i),(t \* (j - 1) + 1):(t \* j), :) = L(i, j, :, :, :);

end

end

## Results

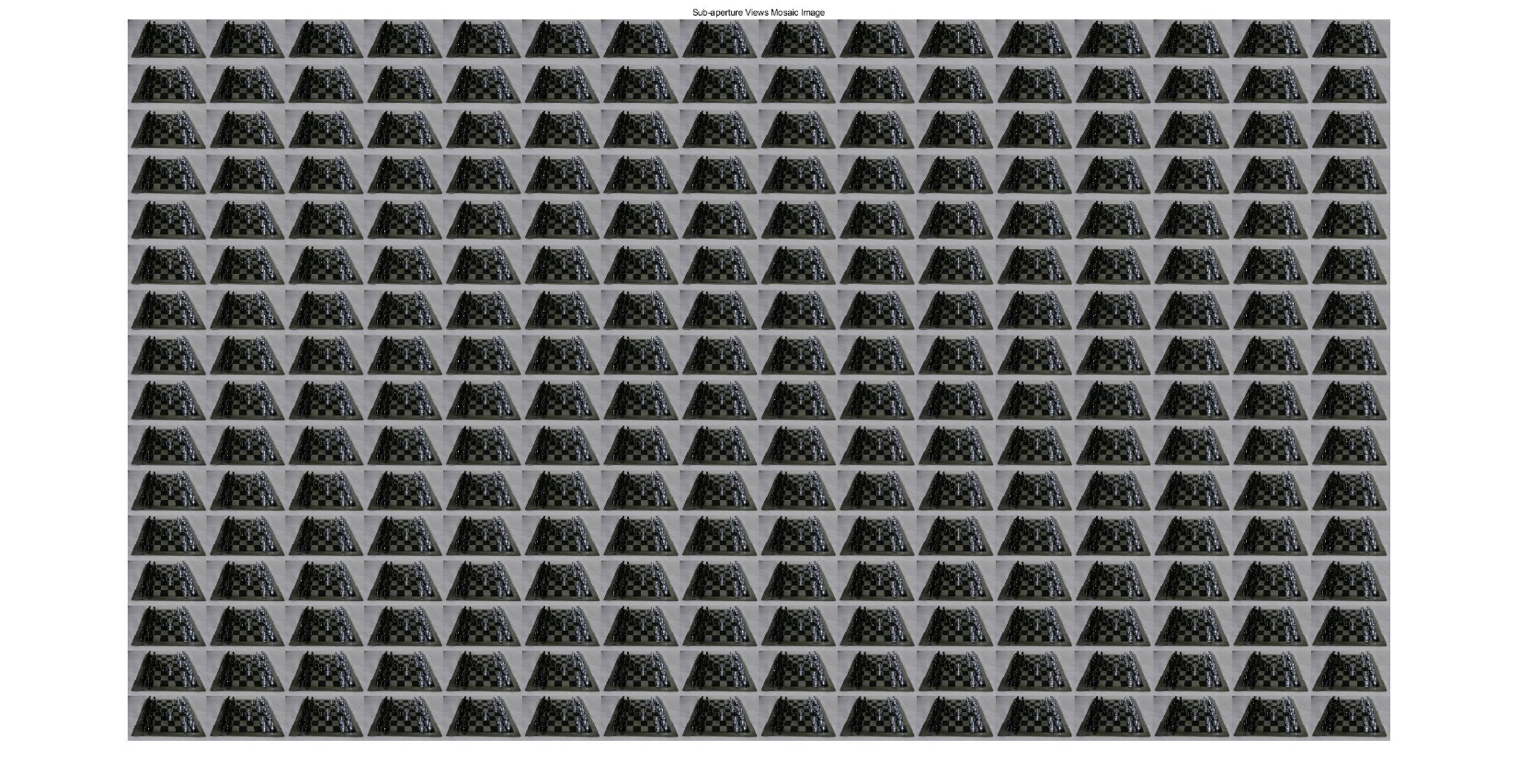


Figure 1. Sub-aperture Views Mosaic Image

# **Refocusing and Focal-stack Generation**

## Description

Figure 2와 같이 Equation(2)을 적용하여 앞서 얻은 sub-aperture 이미지(L)을 합성한다.



Figure 2. Focal Stack Equation

본 보고서에서는 depth 값을 최소 0.0에서 최대 2.0까지 0.2 간격으로 설정하여 총 11장의 refocusing을 실시하였다. 그런데 과제 설명서에 명시된 것처럼 합성 구간을 1에서 16으로 설정한 결과 이미지가 refocusing 뿐만 아니라 약간의 shifting이 발생하는 것으로 보인다. 이에 대한 원인은 찾지 못하였다.

## MATLAB

%% Refocusing and Focal-stack Generation

disp('Refocusing and focal-stack generation..');

for d = 0:0.6:3

img\_refocus = zeros(s, t, c);

for i = 1:s

for j = 1:t

for k = 1:u

for l = 1:v

du = round(i + k \* d);

dv = round(j - l \* d);

img\_refocus(i, j, 1) = img\_refocus(i, j, 1) + L(k, l, du, dv, 1);

img\_refocus(i, j, 2) = img\_refocus(i, j, 2) + L(k, l, du, dv, 2);

img\_refocus(i, j, 3) = img\_refocus(i, j, 3) + L(k, l, du, dv, 3);

end

end

end

end

end

## Results



Figure 3. Image Stack with d = 0.0



Figure 4. Image Stack with d = 0.2



Figure 5. Image Stack with d = 0.4



Figure 6. Image Stack with d = 0.6



Figure 7. Image Stack with d = 0.8



Figure 8. Image Stack with d = 1.0



Figure 9. Image Stack with d = 1.2



Figure 10. Image Stack with d = 1.4



Figure 11. Image Stack with d = 1.6



Figure 12. Image Stack with d = 1.8

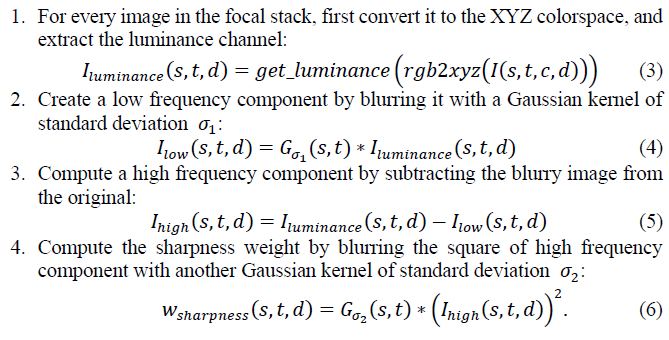


Figure 13. Image Stack with d = 2.0

# **All-focus image and depth from defocus**

## Description

Figure 14에서 제시된 공식들과 같이 이미지의 밝기를 구하기 위해 rgb에서 xyz영역으로의 변환하고 이미지의 low와 high frequency를 통해 blur된 이미지 구하여 Gaussian을 통해 이미지의 sharpness를 구한다.



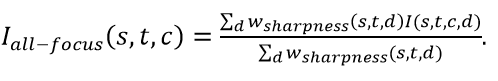


Figure 14. Equational Steps for Defocusing

Low frequency의 standard deviation 값은 2, sharpness weight의 standard deviation 값은 5로 정의한다. 각 standard deviation 값의 변화에 따라 과한 blur가 발생하여 선명도가 떨어짐을 확인하였고 2와 5의 결과 값이 가장 적절한 이미지를 출력하는 것으로 판단된다. Depth map은 아래의 공식과 같이 각 depth 값에 weight를 곱한 후 최대값과의 차를 통해서 구한다.



Figure 15. Depth Map Equation

## MATLAB

%% All-focus Image and Depth from Defocus

disp('All-focus image and depth from defocus..');

[row, col] = size(img\_stack);

std\_1 = 2;

std\_2 = 5;

for i = 1:col

img\_lum{end + 1} = rgb2xyz(img\_stack{i}, 'ColorSpace', 'srgb');

img\_low{end + 1} = imgaussfilt(img\_lum{end}, std\_1);

img\_high{end + 1} = img\_lum{end} - img\_low{end};

img\_sharp{end + 1} = imgaussfilt(img\_high{end} .^ 2, std\_2);

end

for i = 1:s

for j = 1:t

for k = 1:col

img\_defocus(i, j, 1) = img\_defocus(i, j, 1) + ... img\_stack{k}(i, j, 1) \* img\_sharp{k}(i, j);

img\_defocus(i, j, 2) = img\_defocus(i, j, 2) + ...

img\_stack{k}(i, j, 2) \* img\_sharp{k}(i, j);

img\_defocus(i, j, 3) = img\_defocus(i, j, 3) + ...

img\_stack{k}(i, j, 3) \* img\_sharp{k}(i, j);

weight = weight + img\_sharp{k}(i, j);

img\_dmap(i, j) = img\_dmap(i, j) + img\_sharp{k}(i, j) \* k;

end

img\_defocus(i, j, :) = img\_defocus(i, j, :) / weight;

img\_dmap(i, j) = img\_dmap(i, j) / weight;

end

end

img\_dmap(i, j) = 1. - img\_dmap(i, j) / col;

## Results

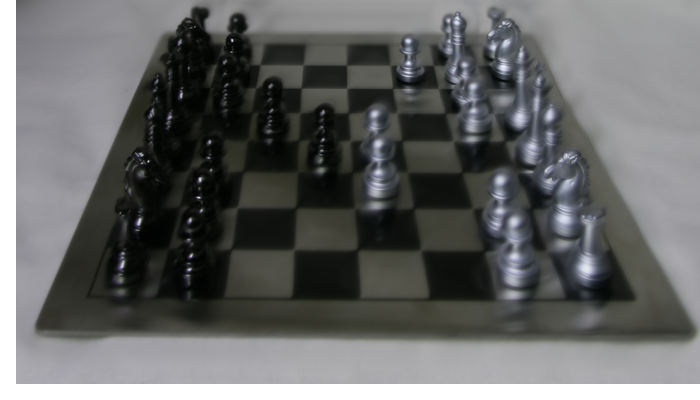


Figure 16. Defocused Image

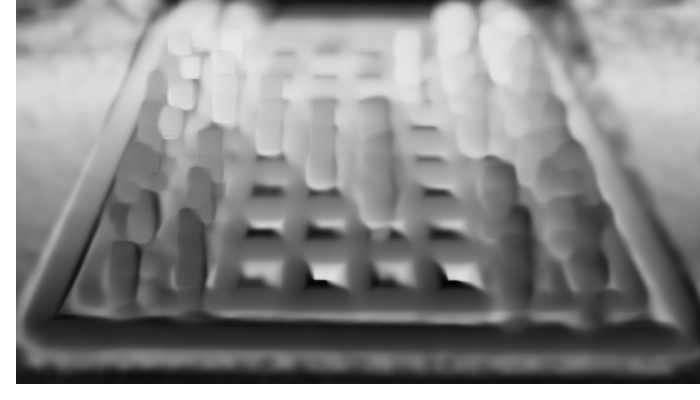


Figure 17. Depth Map Image