**Q1. Initials & Q2. Sub-aperture views**

**Description**

주어진 plenoptic 카메라 이미지가 1픽셀당 16x16의 aparture domain source를 가지고 있으므로, 이 lightfield의 (u,v)차원은 16x16이 된다.

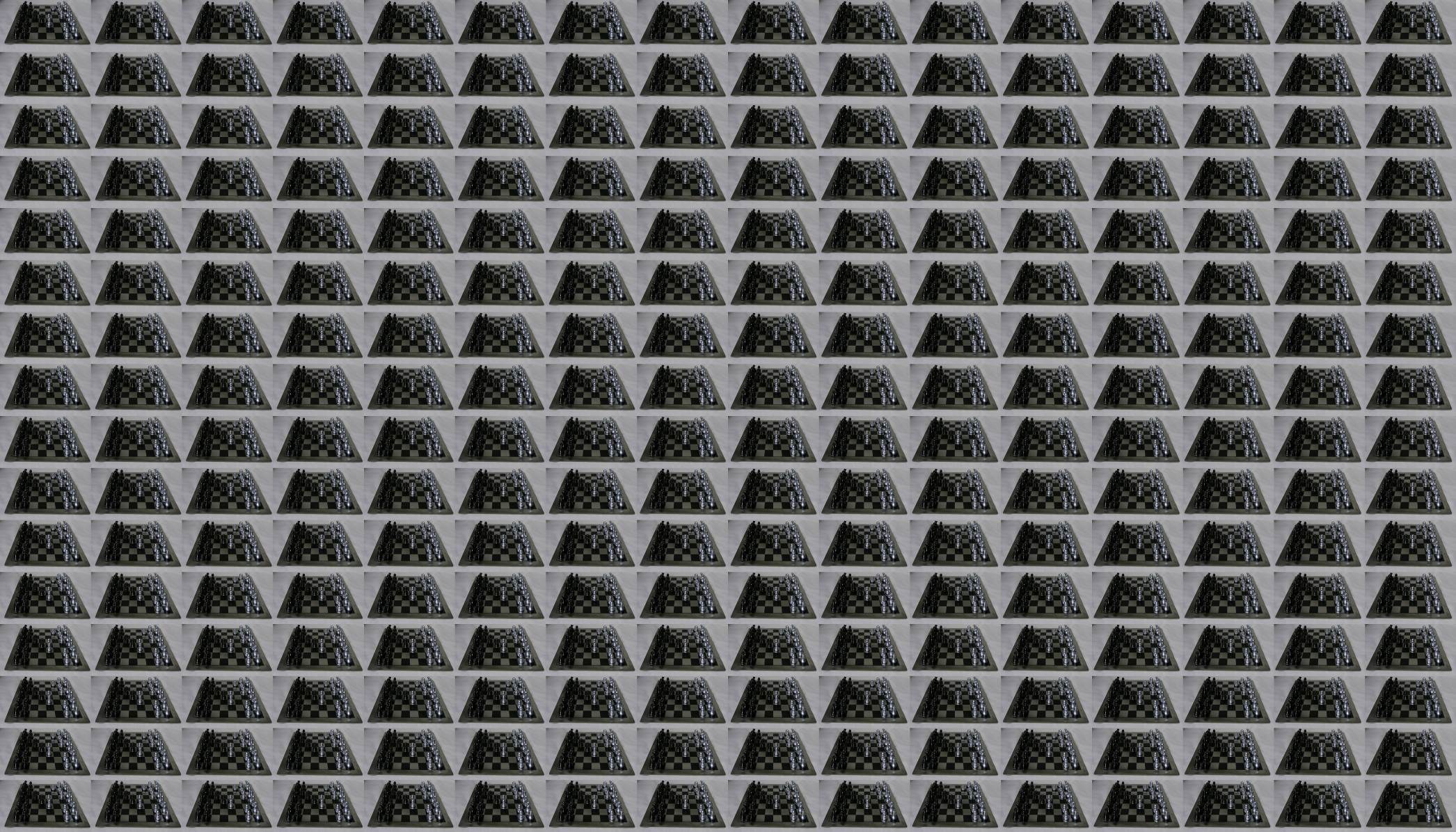
따라서 L = 16x16x400x700x3 matrix를 정의하고 여기에 이미지의 픽셀을 채워 넣으면 된다.

그리고 mosaic 이미지는 L(x, y, :, :, :)가 그대로 aparture domain에서 x, y점의 pinhole image가 된다.

**Code**

|  |
| --- |
| img\_plenoptic = imread('data/chessboard\_lightfield.png');  lightfield = zeros(16, 16, 400, 700, 3);  for x = 1:400  for y = 1:700  for i = 1:16  for j = 1:16  for c = 1:3  lightfield(i, j, x, y, c) = double(img\_plenoptic( (x - 1) \* 16 + i, ...  (y - 1) \* 16 + j, c)) / 256.;  end  end  end  end  end  img\_mosaic = zeros(6400, 11200, 3);  for i = 1:16  for j = 1:16  img\_mosaic( (i-1) \* 400 + 1 : (i-1) \* 400 + 400, (j-1) \* 700 + 1 : (j-1) \* 700 + 700, :) ...  = lightfield(i, j, :, :, :);  end  end  imwrite(img\_mosaic, 'mosaic.png'); |

**Result**



원본 이미지 링크 : https://boratw.github.io/Homeworks/IIT6028/hw5/mosaic.png

**Q3. Refocusing and focal-stack generation**

**Description**

공식을 그대로 적용하였다. 이때 u,v의 범위가 [1,16]이 아닌 [-7,8]이어야 한다.

또한 어째서인지 matlab에서는 y축 방향을 반대로 해야 잘 적용되었다. 아마 matlab 내부의 이미지 처리 방법과 관련이 있는 것으로 보인다.

**Code**

|  |
| --- |
| img\_depth = {};  for d = 0:0.1:2  img = zeros(400, 700, 3);  for x = 1:400  for y = 1:700  sum = 0;  for i = -7:8  for j = -7:8  sx = round(x + i \* d);  sy = round(y - j \* d);  if sx >= 1 && sx <= 400 && sy >= 1 && sy <= 700  for c = 1:3  img(x, y, c) = img(x, y, c) + lightfield(i+8, j+8, sx, sy, c);  end  sum = sum + 1;  end  end  end  img(x, y, :) = img(x, y, :) / sum;  end  end  imwrite(img, strcat('img\_depth\_',num2str(d),'.png'));  img\_depth{end + 1} = img;  end |

**Result**

|  |  |  |
| --- | --- | --- |
| d = 0 | d = 0.6 | d = 1.2 |
|  |  |  |

**Q4. All-focus image and depth from defocus**

**Description**

문제에 주어진 대로 각 depth 이미지의 밝기를 구하고 여기에서 이를 blur시킨 이미지를 뺌으로서 이미지의 sharpness를 구하였다.

이후 sharpness를 weight으로 하여 전체 depth 이미지들을 합성하였다.

low frequency 이미지를 구하기 위한 gaussian filter의 stddev는 2,

sharpness를 blurring하기 위한 gaussian filter의 stddev는 5를 사용하였다.

다음으로 depth map을 구하기 위해 각 depth값에 위의 weight를 동일하게 주어서 합성하였다.

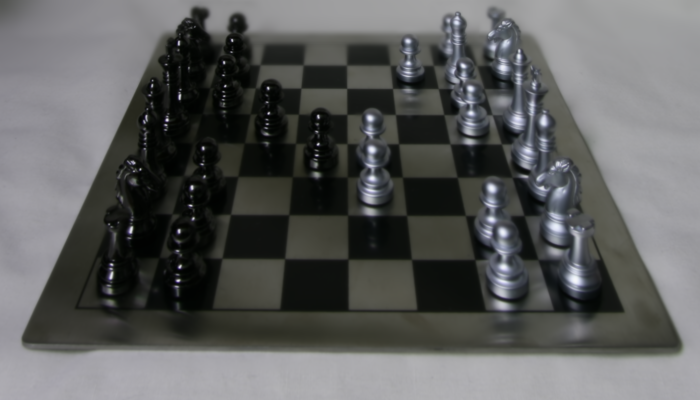
값이 작을수록 더 먼 물체이므로 depth의 최대값으로 나눈 후 1에서 빼면 상대적 깊이를 구할 수 있다.

**Code**

|  |
| --- |
| img\_depth\_lum = {};  img\_depth\_low = {};  img\_depth\_high = {};  img\_depth\_sharp = {};  for d = 1:21  % Get luminance  img\_depth\_lum{end + 1} = rgb2xyz(img\_depth{d}, 'ColorSpace', 'srgb');  img\_depth\_lum{end} = img\_depth\_lum{end}(:, :, 2);  % Get low frequency  img\_depth\_low{end + 1} = imgaussfilt(img\_depth\_lum{end}, 2);  % Get high frequency  img\_depth\_high{end + 1} = img\_depth\_lum{end} - img\_depth\_low{end};  % Get sharpness  img\_depth\_sharp{end + 1} = imgaussfilt(img\_depth\_high{end} .^ 2, 5);  end  img\_all\_focus = zeros(400, 700, 3);  img\_depth\_gray = zeros(400, 700);  for x = 1:400  for y = 1:700  w = 0;  for d = 1:21  for c = 1:3  img\_all\_focus(x, y, c) = img\_all\_focus(x, y, c) + img\_depth{d}(x, y, c) \* img\_depth\_sharp{d}(x, y);  end  img\_depth\_gray(x, y) = img\_depth\_gray(x, y) + d \* img\_depth\_sharp{d}(x, y);  w = w + img\_depth\_sharp{d}(x, y);  end  img\_all\_focus(x, y, :) = img\_all\_focus(x, y, :) / w;  img\_depth\_gray(x, y) = img\_depth\_gray(x, y) / w;  end  end  imwrite(img\_all\_focus, 'img\_all\_focus.png');  imwrite(1. - img\_depth\_gray / 21, 'img\_depth.png'); |

**Result**

All-focused Image



Depth map

