Assignment 4

Anirudha Ramesh (aramesh3)

November 7, 2022

Please look at the code submitted along-side. There are numerous comments describing what we are doing and answering/elaborating on some answers as well as having the necessary visualizations.

1. Part 1 : Lightfield rendering, depth from focus, and confocal stereo (100 points)

   (a) Sub-aperture views

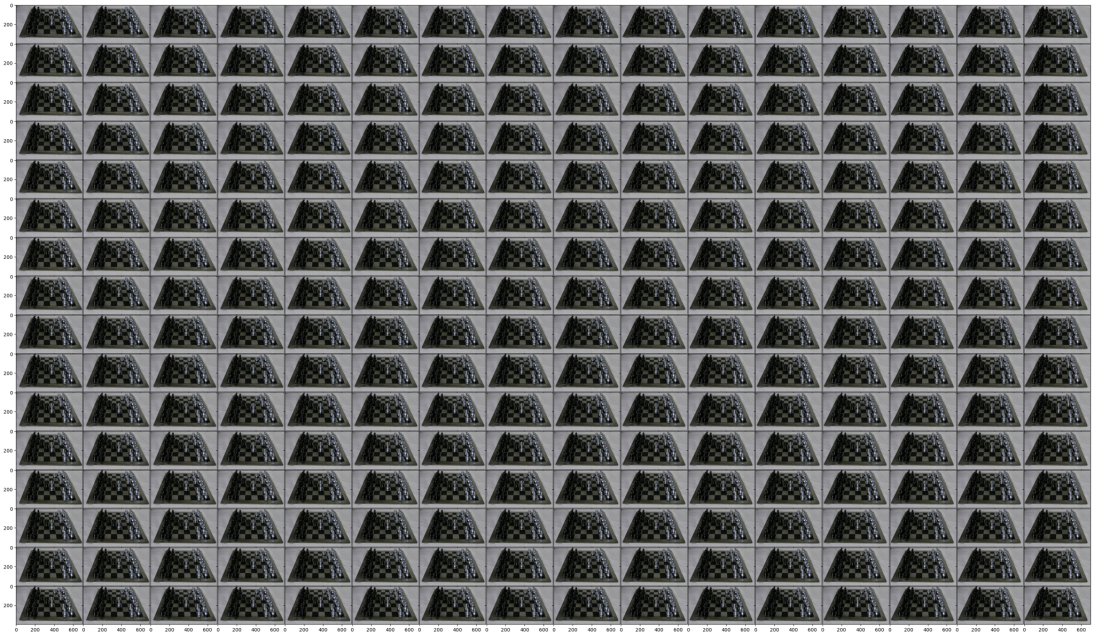

Figure 1: Sub Aperture Stack produced

(b) Refocusing and Focal Stack Simulation

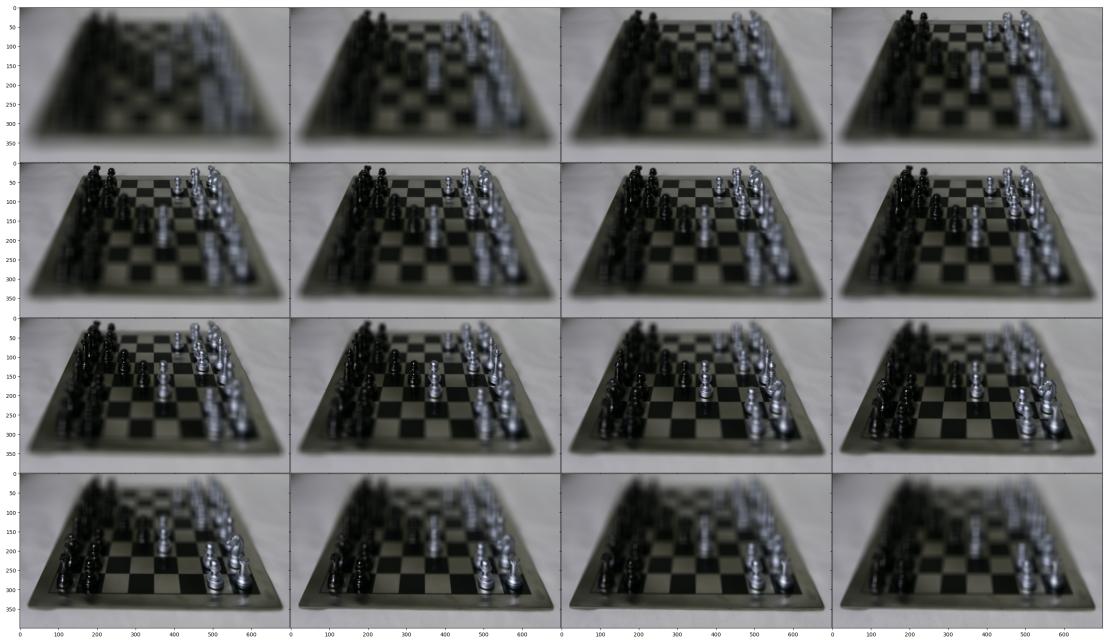

Figure 2: Different Refocusings

(c) All in-focus image and depth from focus

Areas lacking texture are one prominent failure case of our depth from focus. This can be seen in the center of individual chessboard tiles towards the bottom. This is hard to make out in the all-in-focus image as this area has no texture. This is also visible towards the edges of the board, but this on the other hand is slightly hazy, particularly towards the bottom, but overall in most textured parts, we have good performance. Since this isn't per pixel we tend have chunks assigned to the same depth, thus potentially giving us generally blurry images in regions with rapid change of depth. In our image this scenario isn't present, and so we don't have much to worry about.

Varying sigma1 and sigma2 does not make a very big difference on how we visually see the all-in-focus image. However, their effects on the depth-map are more visible.

With increase in sigma1 there is little change, but there is an increase in the 'glow' (halo) around objects in the depth map. This is as expected with more blurring the low freq image differs more than the high-freq image. We choose to pick the sigma1 = 50 image just because it offers a balance between the either end.

With increase in sigma2 we can see that though the edges in the depth map become smoother, the finer depth details, particularly towards the back disappear. This is a positive in case of the chessboard tiles with no texture, but not in the case of pieces towards the back of the board. We choose to keep sigma2 low based on wanting clarity in the latter cases, even if it comes at the cost of performance on the former.

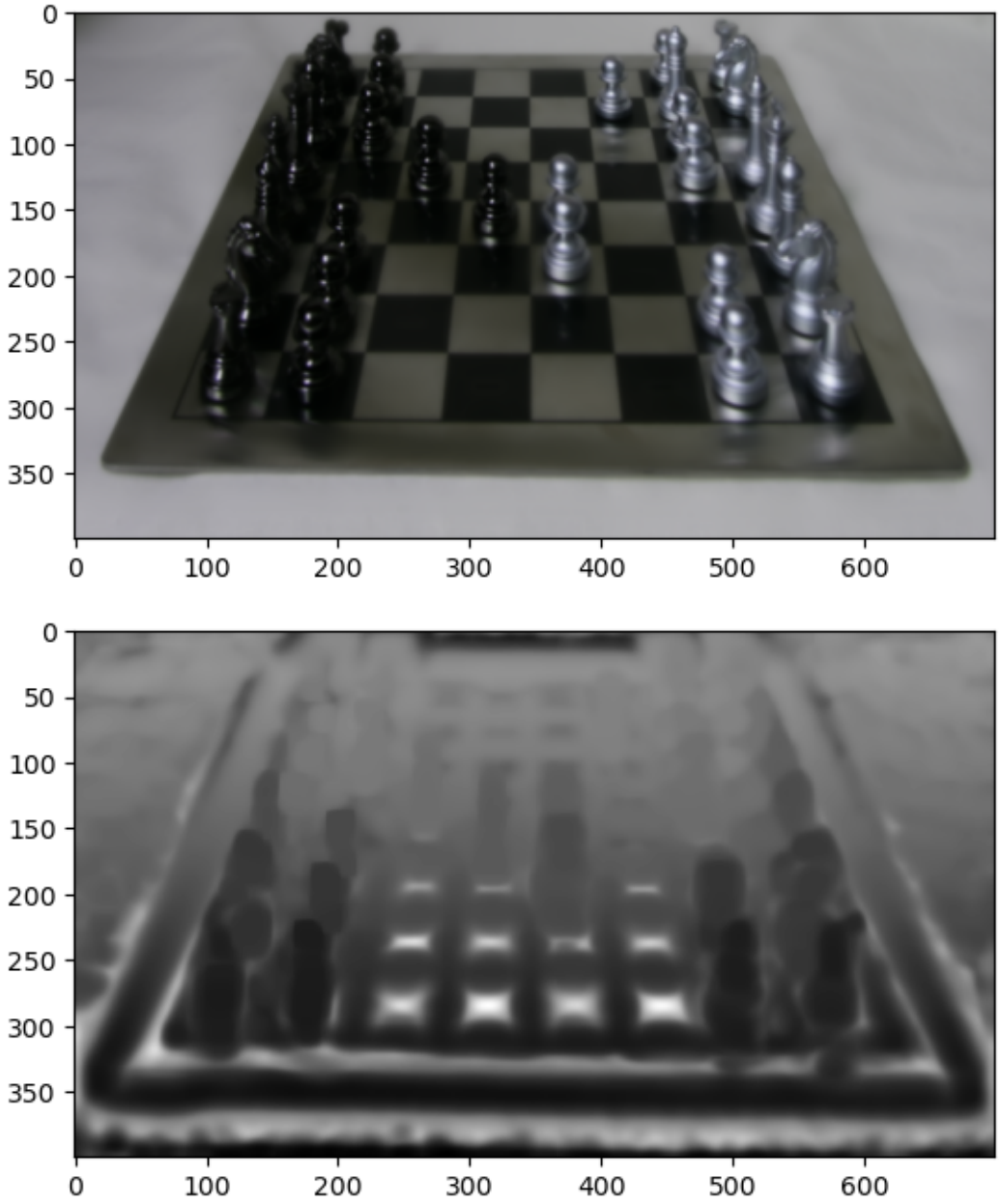

Figure 3: All in-focus image and depth from focus images for sigma1 = 50 and sigma2 = 5

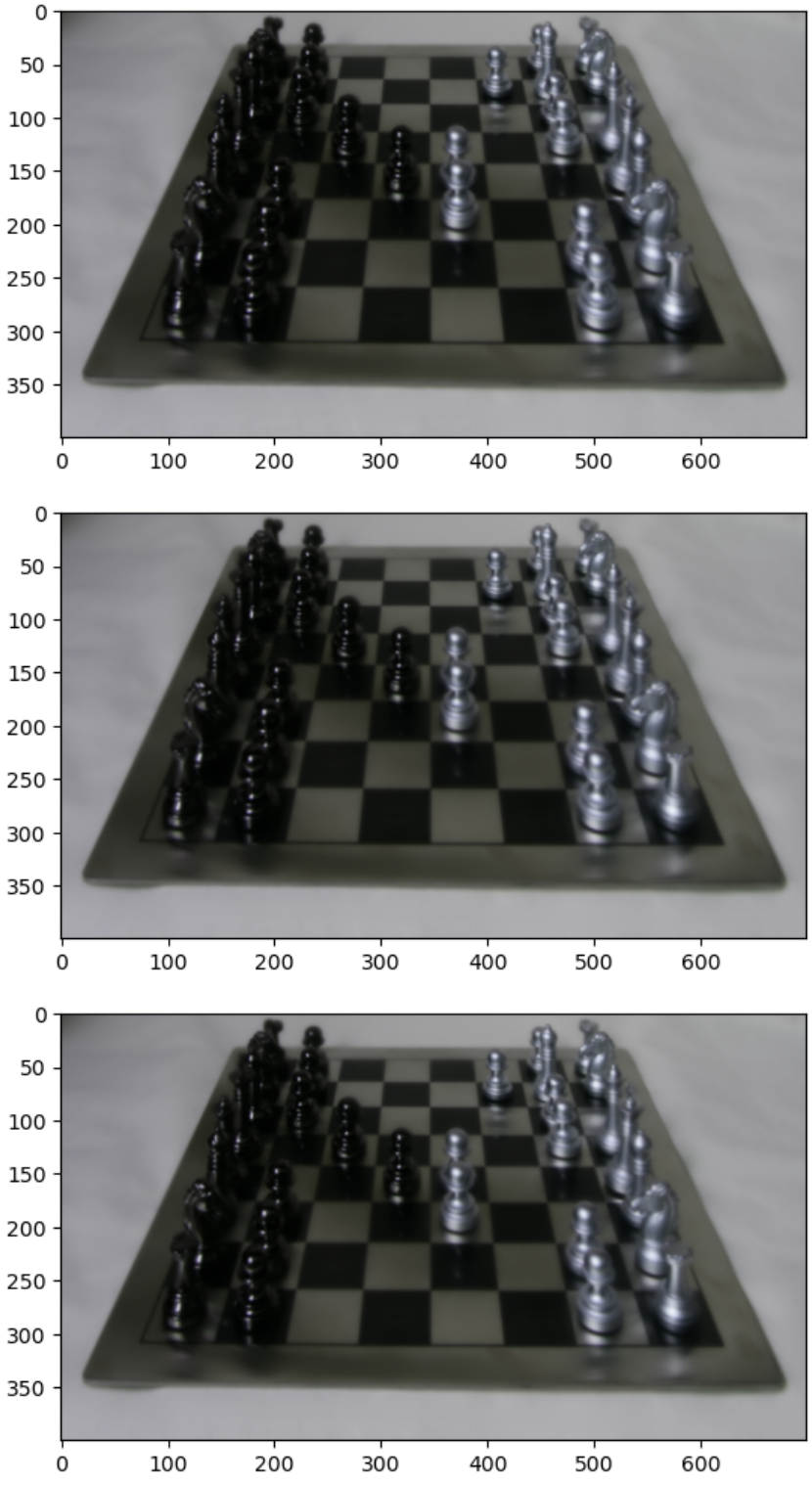

Figure 4: varying sigma1 : 5, 50, 500 (sigma2 = 5)

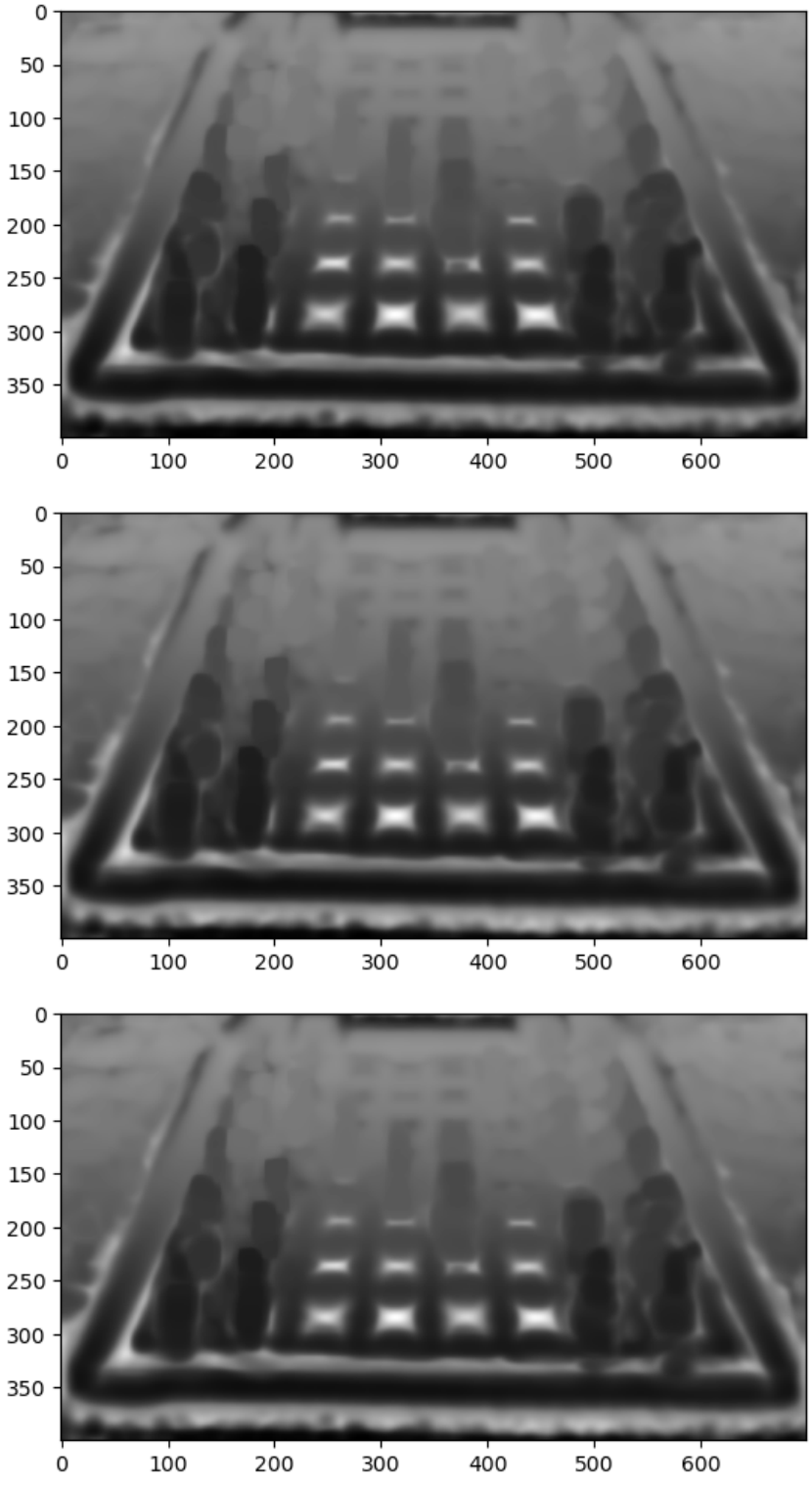

Figure 5: varying sigma1 : 5, 50, 500 (sigma2 = 5)

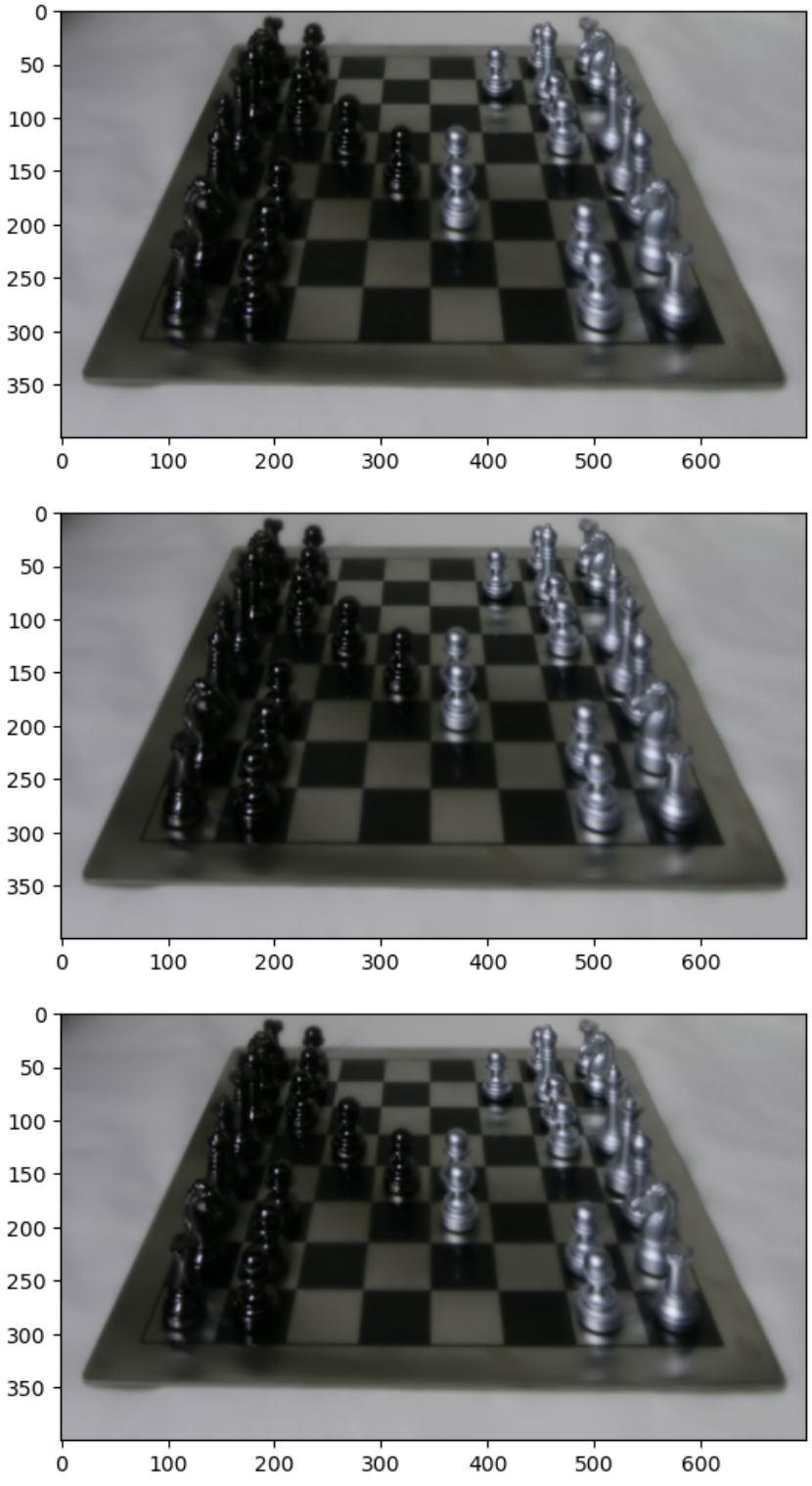

Figure 6: varying sigma2 : 5, 50, 500 (sigma1 = 50)

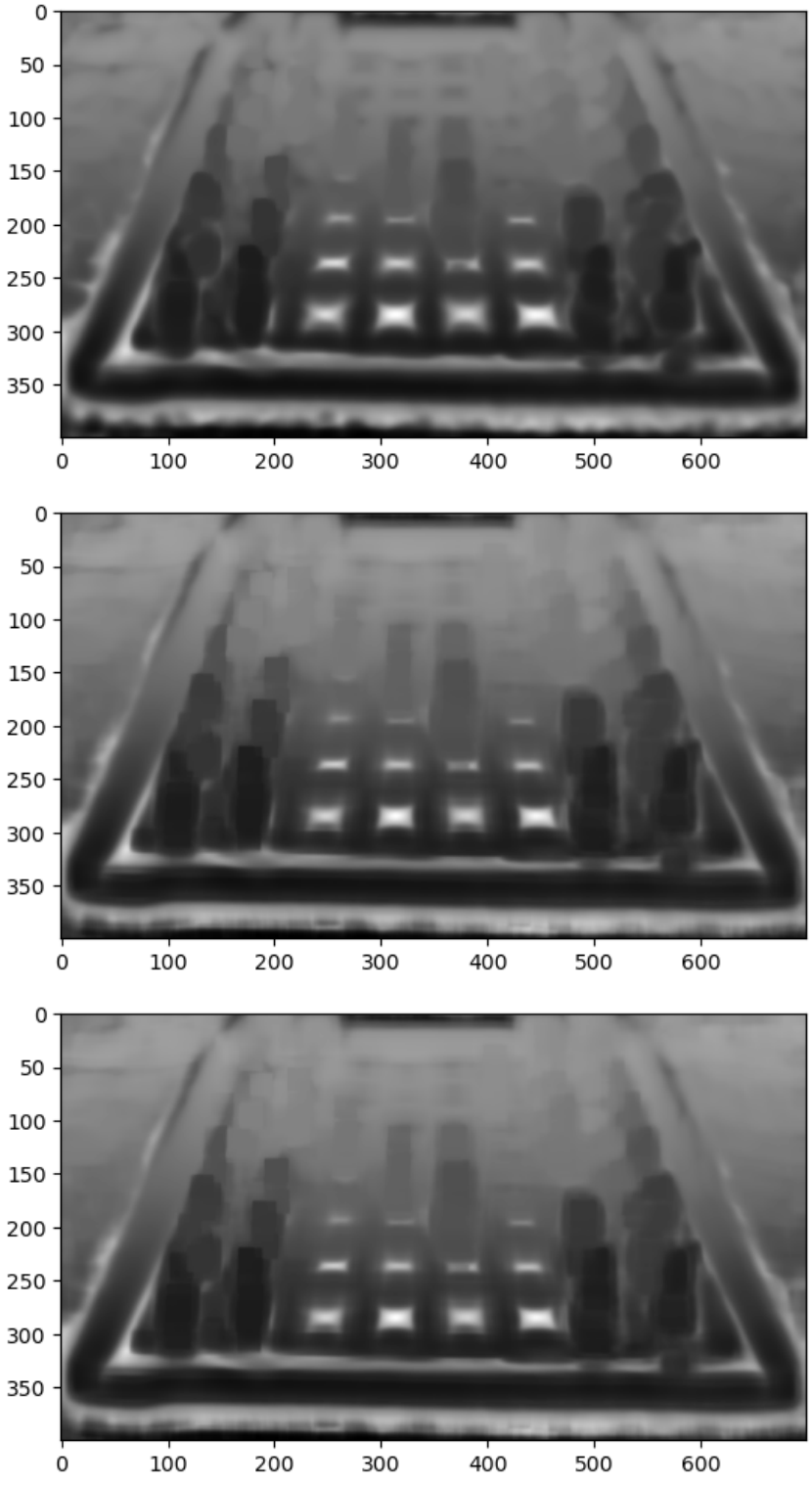

Figure 7: varying sigma2 : 5, 50, 500 (sigma1 = 50)

(d) Focal-aperture stack and confocal stereo

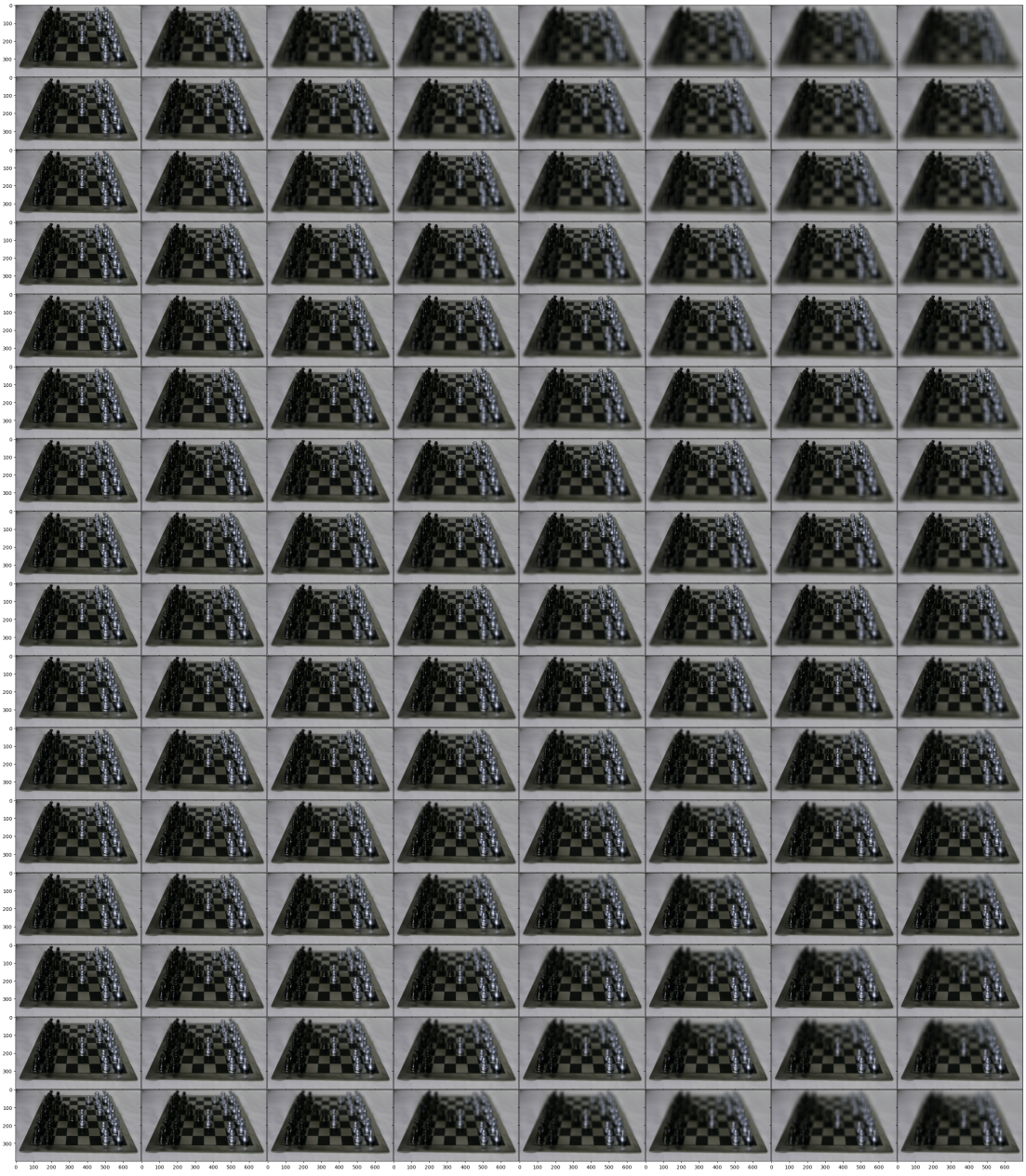

Figure 8: focal aperture stack. aperture increases as we go right, and and focal plane comes closer as we go down.

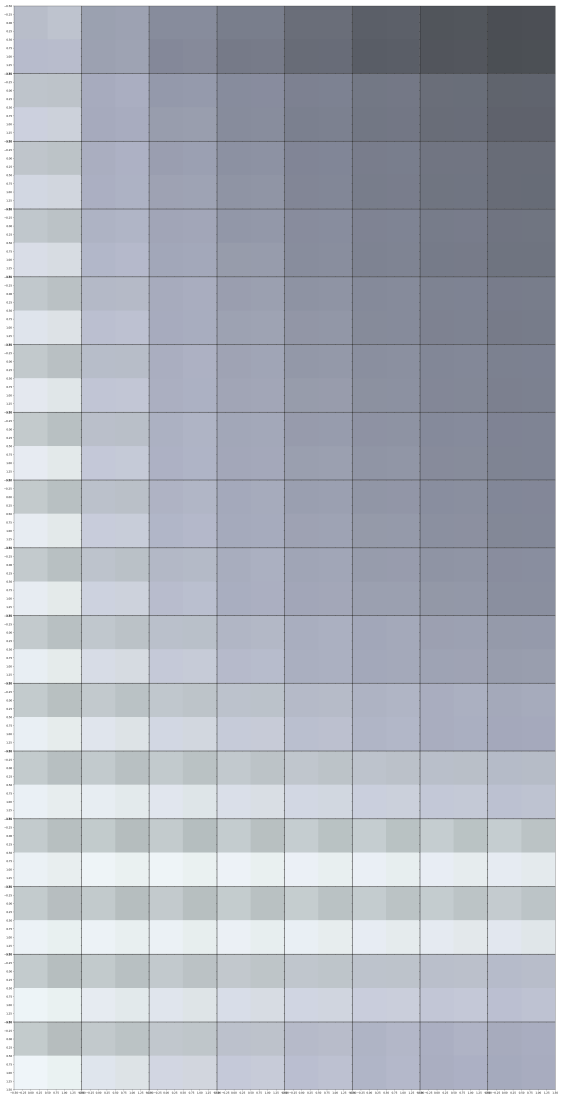

Figure 9: AFI for a section of 2*2 pixels to get an idea of how confocal consistency works, as well as relationships for pixels in the same locality.

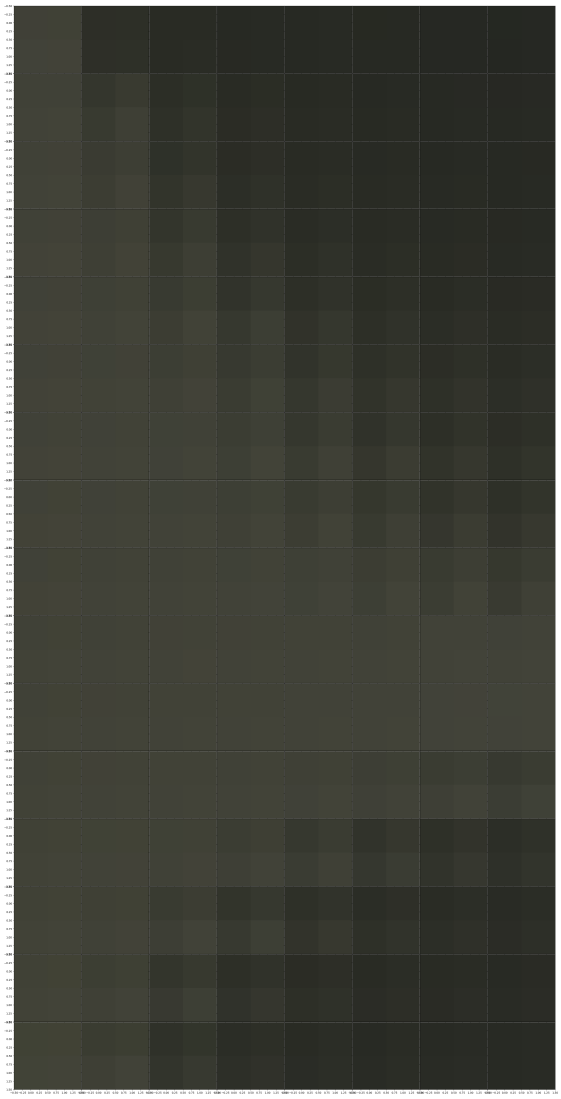

Figure 10: AFI for a section of 2*2 pixels to get an idea of how confocal consistency works, as well as relationships for pixels in the same locality.

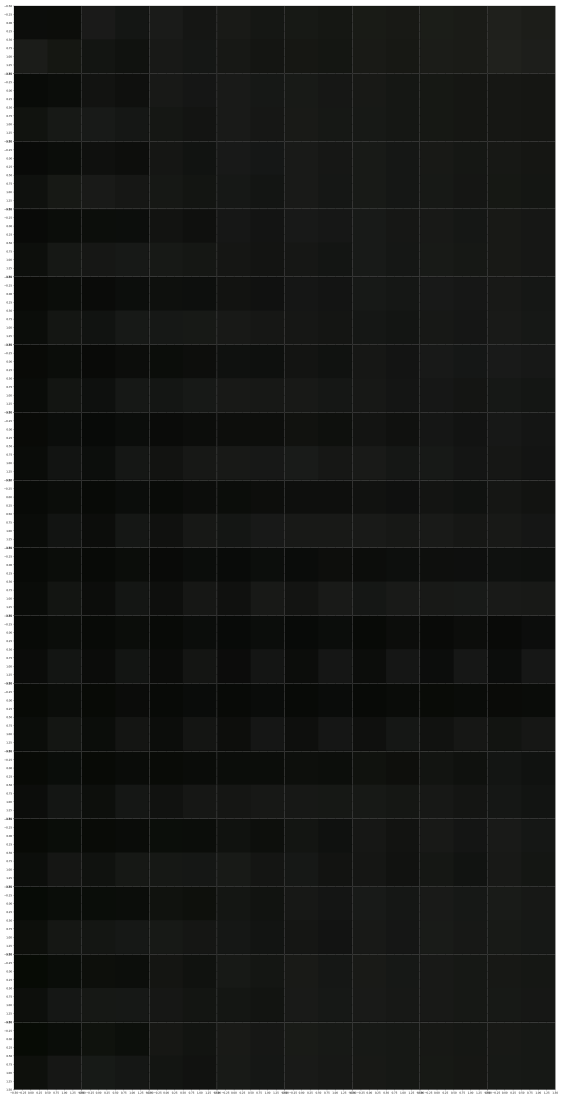

Figure 11: AFI for a section of 2*2 pixels to get an idea of how confocal consistency works, as well as relationships for pixels in the same locality.

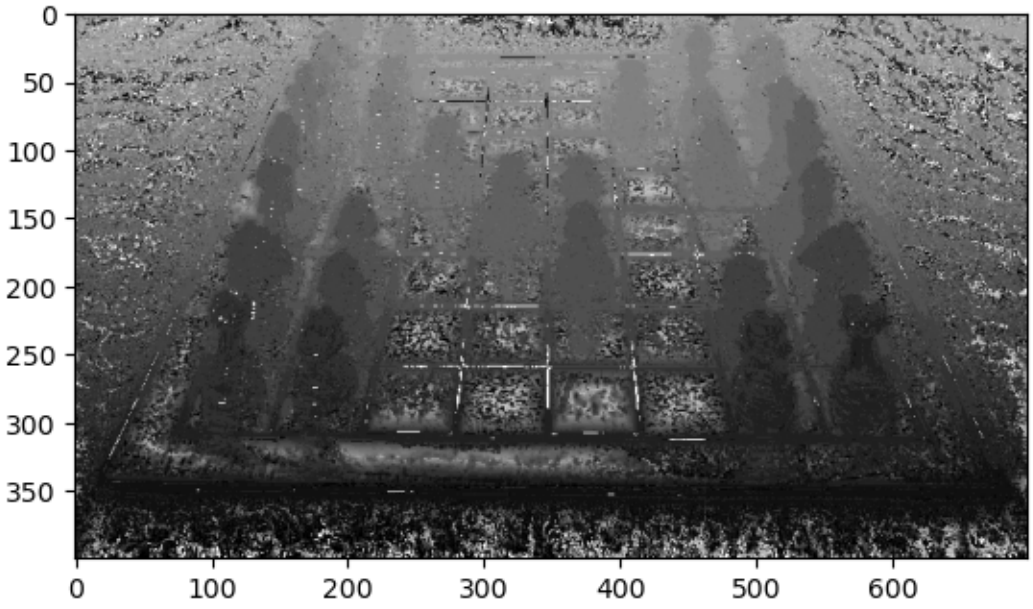

Figure 12: Depth from confocal. The depth map in this case is calculated on a per pixel basis, and so captures a lot of finer depth detail the other case misses out on. But at the same time, without a regularizer, the depth map isn't very continuous. The depth from focus procedure was not per-pixel in the same way, as the sharpness metric requires neighborhood estimates, but as a consequence of this, its naturally smoother and more continuous.

2. Capture and Refocus your own lightfield (100 points)

   (a) Capturing your own light-field.

   In our selection we balance out number of frames and down-sampling to get a good tradeoff between performance and timing.
   We find that performance does not vary too much even if we don't take too many frames (50-100) or go higher(250-300).

   The video used in a part of the submitted github repo in the folder data-dump.

   (b) Refocusing an unstructured lightfield

   We implement eq9 pixel-wise on our own. Since we are doing this pixel-wise on our own, we don't have to use any scipy functions. This also makes it easy to create I_t_bar. It is also clear that subtraction of mean/I_bar is mainly required for the normalization term.

```
def get_normxcorr(img, template, bbox) :
    d1 = np.sum((template - template.mean())**2)
    normxcorr = np.zeros(img.shape)

    bbox_center = [(bbox[0] + bbox[2])/2, (bbox[1]+bbox[3])/2]
    bbox_size = np.array(template.shape) * 2
    count = 0

    for i in range(img.shape[0]) :

        if np.abs(i - bbox_center[0]) > bbox_size[0] : continue

        for j in range(img.shape[1]) :

            if np.abs(j - bbox_center[1]) > bbox_size[1] : continue

            if img[i:i+template.shape[0], j:j+template.shape[1]].shape != template.shape : continue

            count += 1
            num = np.sum(img[i:i+template.shape[0], j:j+template.shape[1]] * template)
            I_box = img[i:i+template.shape[0], j:j+template.shape[1]] / img[i:i+template.shape[0],
j:j+template.shape[1]].sum()
            d2 = np.sum((img[i:i+template.shape[0], j:j+template.shape[1]] - I_box) ** 2)
            normxcorr[i,j] = num / np.sqrt(d1*d2)

    return normxcorr

def calculate_shift(normxcorr, bbox) :

    y, x = np.unravel_index(np.argmax(normxcorr), normxcorr.shape)

    y_mid_im, x_mid_im = bbox[0], bbox[1]

    shift_x = x - x_mid_im # col shift
    shift_y = y - y_mid_im # row shift

    return shift_x, shift_y
```

Figure 13: Normalized Cross Correlation, and matching implementation

The jupyter notebook for q2, and the pdf of the jupyter notebook show the tracking success, which it seems to always get right.
Below are some different focuses.

Figure 14: Template set to the pig, the mug, and the chips in the back. The template sizes are around 60x60 - 100x100 pixels.