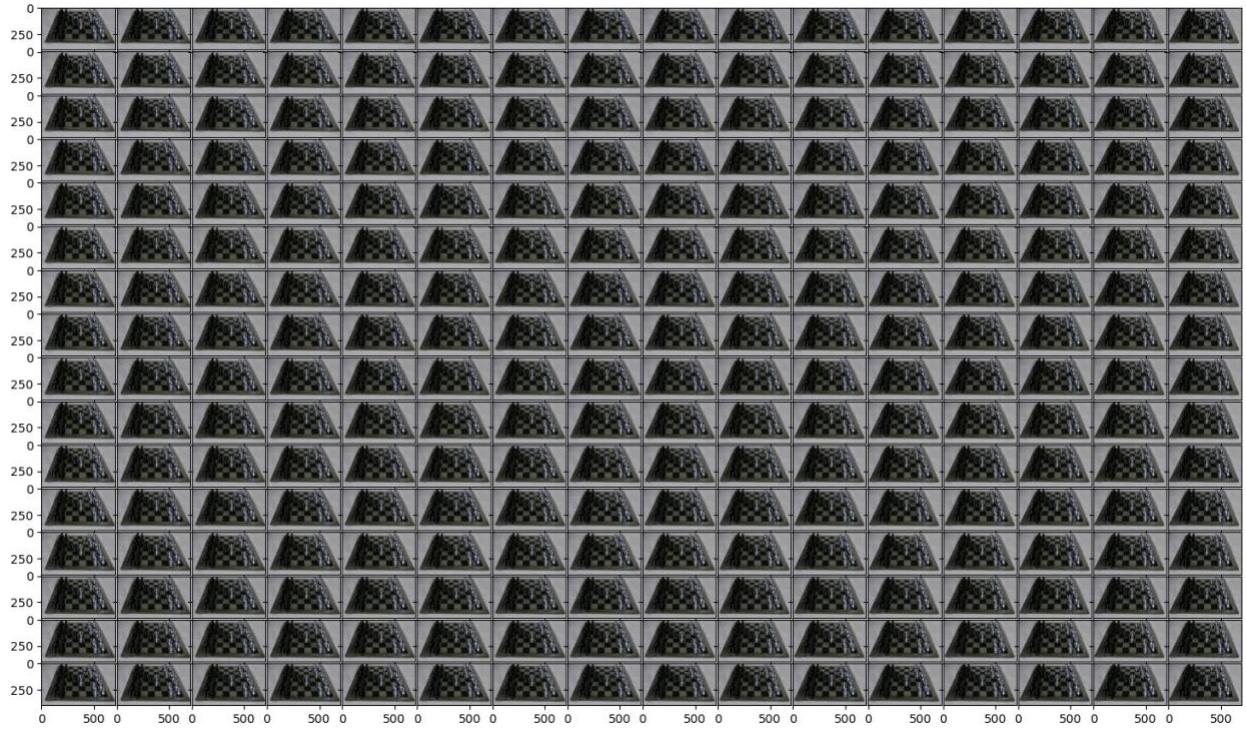
15-463 Computational Photography (Fall 2023)

Assignment 4

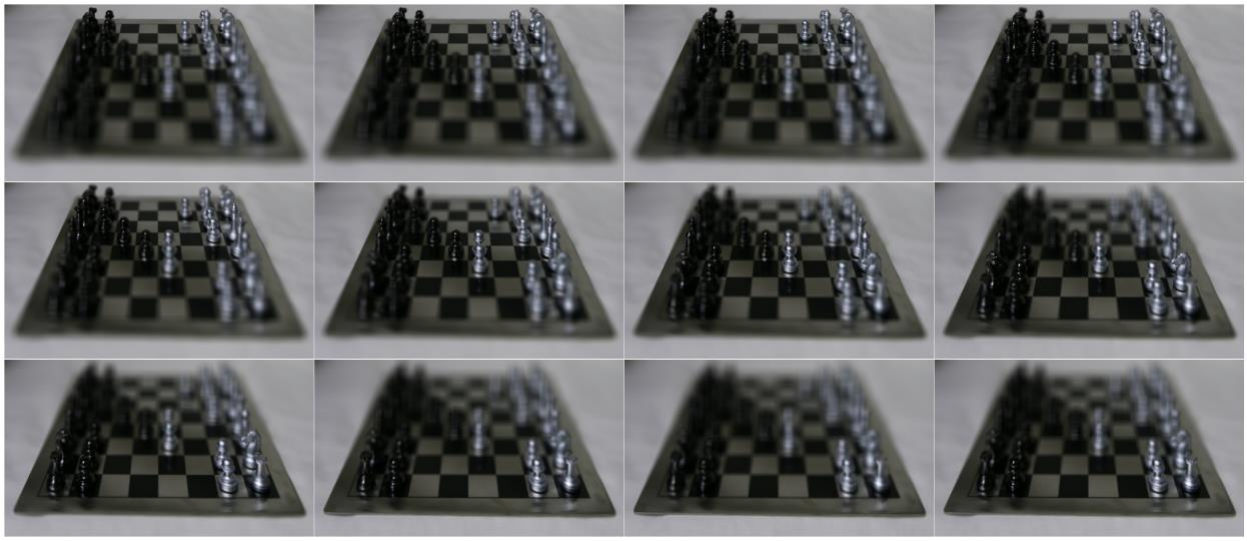
Haejoon Lee

1. Lightfield rendering, depth from focus, and confocal stereo

[Sub-aperture views]



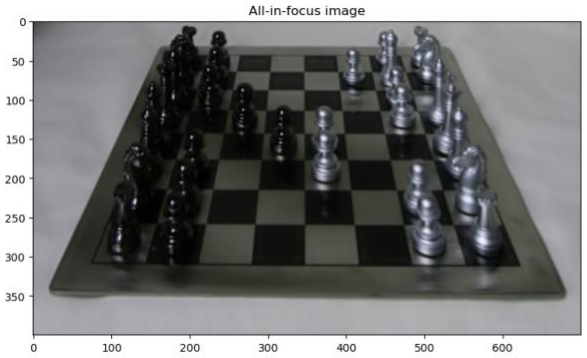
[Refocusing and focal-stack simulation]

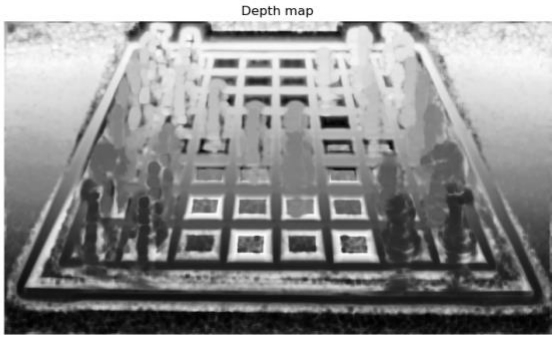


[All-in-focus image and depth from focus]

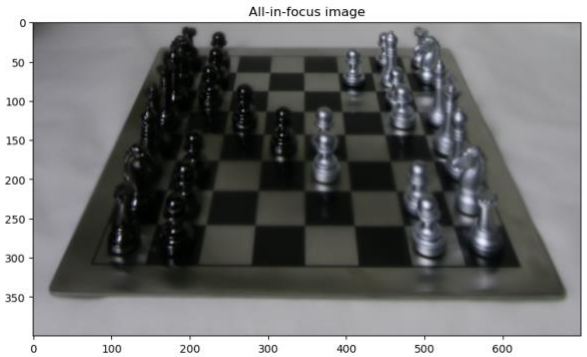
1) Changing blur kernel size

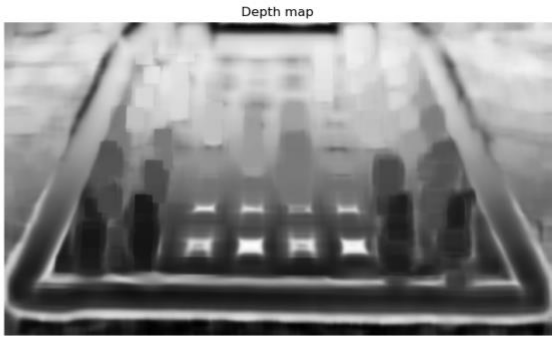
Fixed sigma 1: 20, sigma2: 20



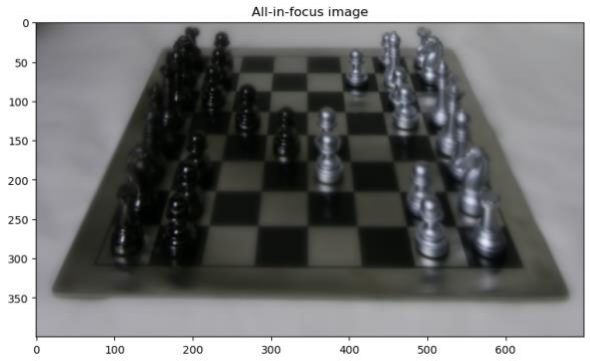


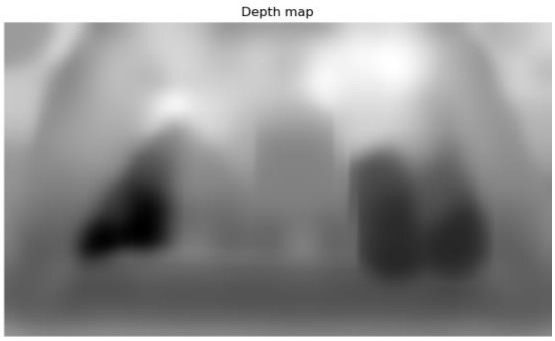
Kernel size: [21, 21]





Kernel size: [99, 99]

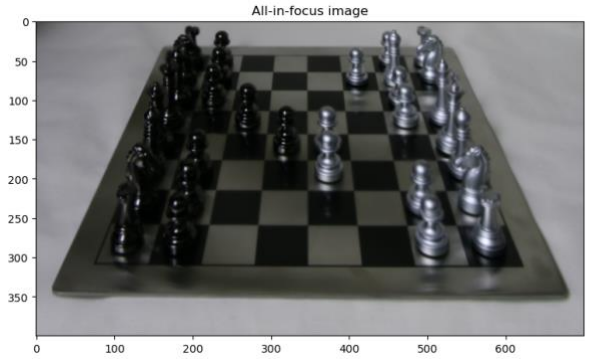


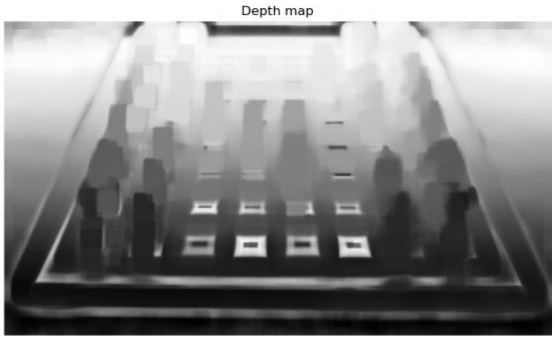


2) Changing sigma 1

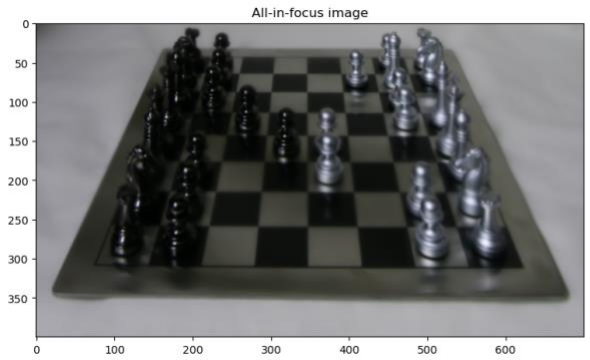
Fixed kernel size: [21, 21], sigma2: 20

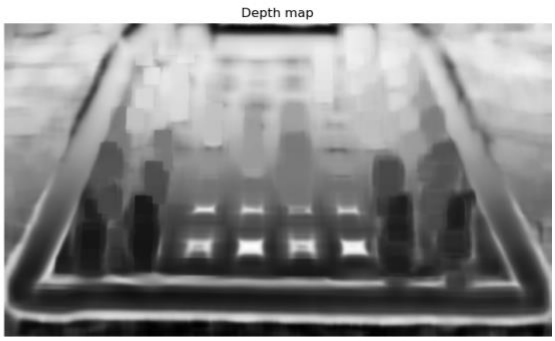
Sigma1: 1



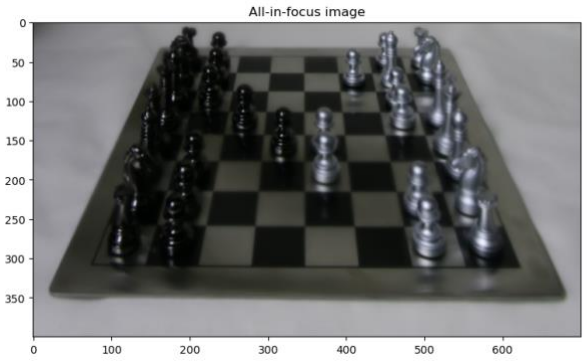


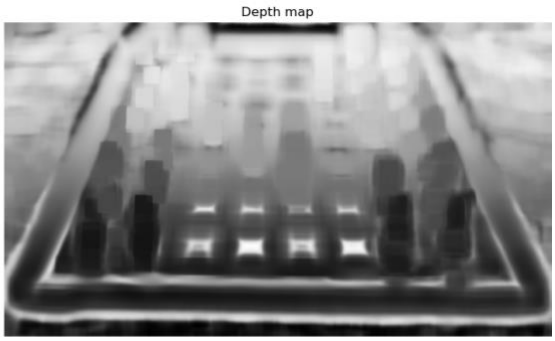
Sigma1: 20





Sigma1: 100

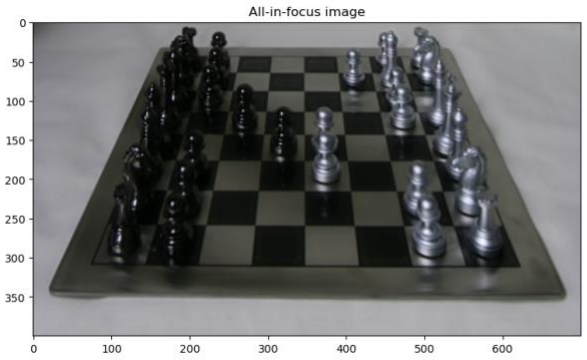


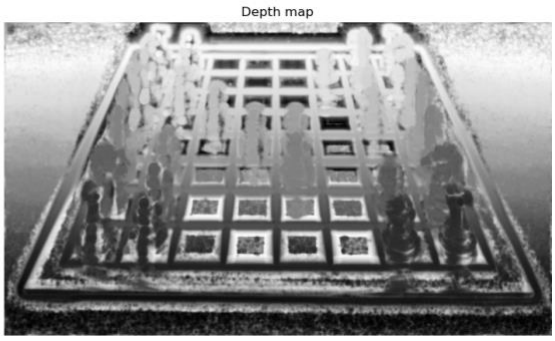


3) Changing Sigma 2

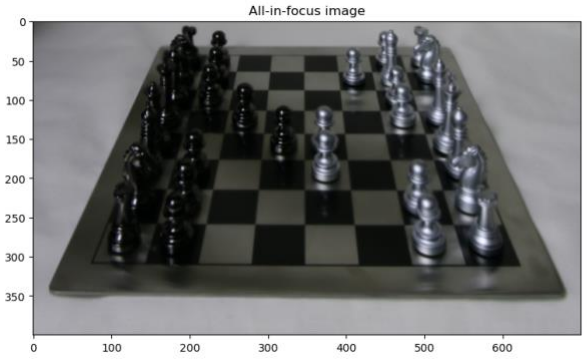
Fixed kernel size: [21, 21], sigma1: 1

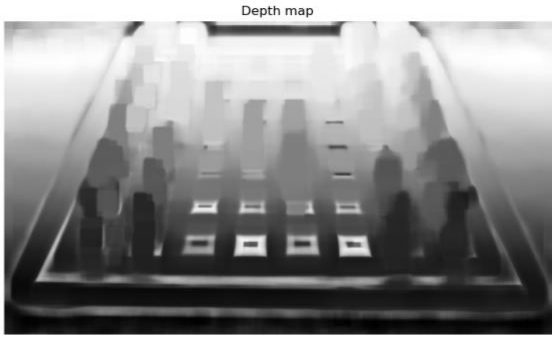
Sigma2: 1



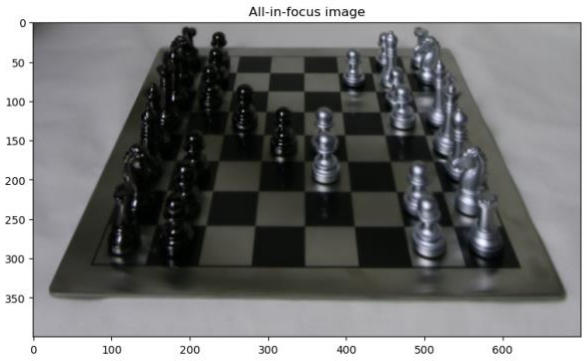


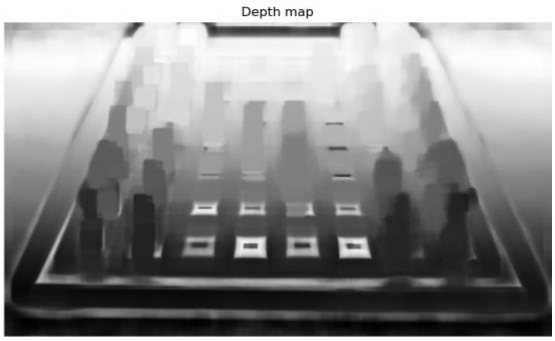
Sigma2: 20





Sigma2: 100



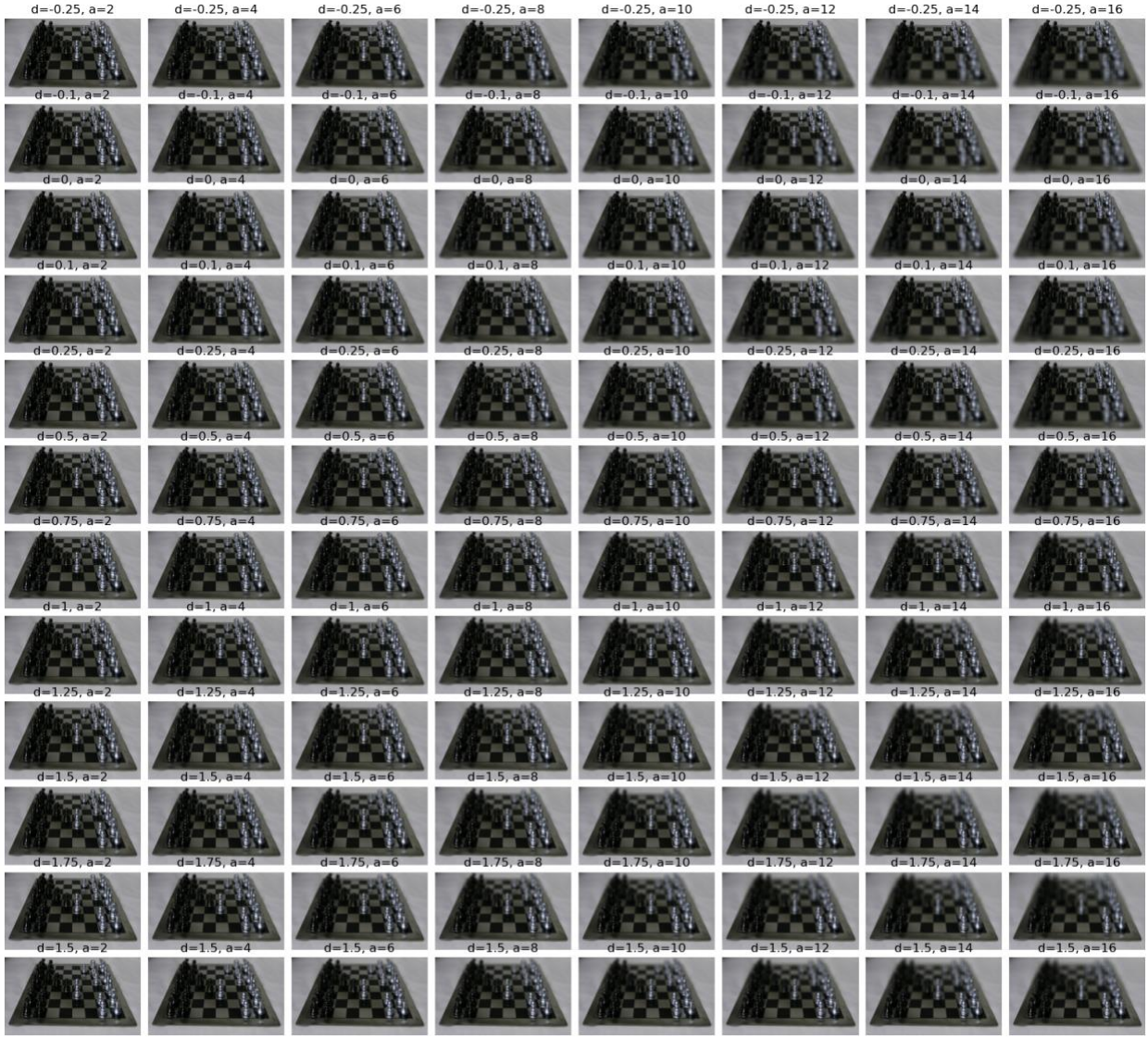


Best parameter: kernel size = [21, 21], Sigma1 = 1, Sigma2 = 20

Depth map was incorrect within the smooth part (especially empty board) because they don't have details for inference of depth based on the change of focus. Another part with inaccurate estimated depth is the edge of the board in front and behind, which is blurrier than the other parts in All-in-Focus image. Probably the incorrect depth estimation affected the image and resulted in the blur.

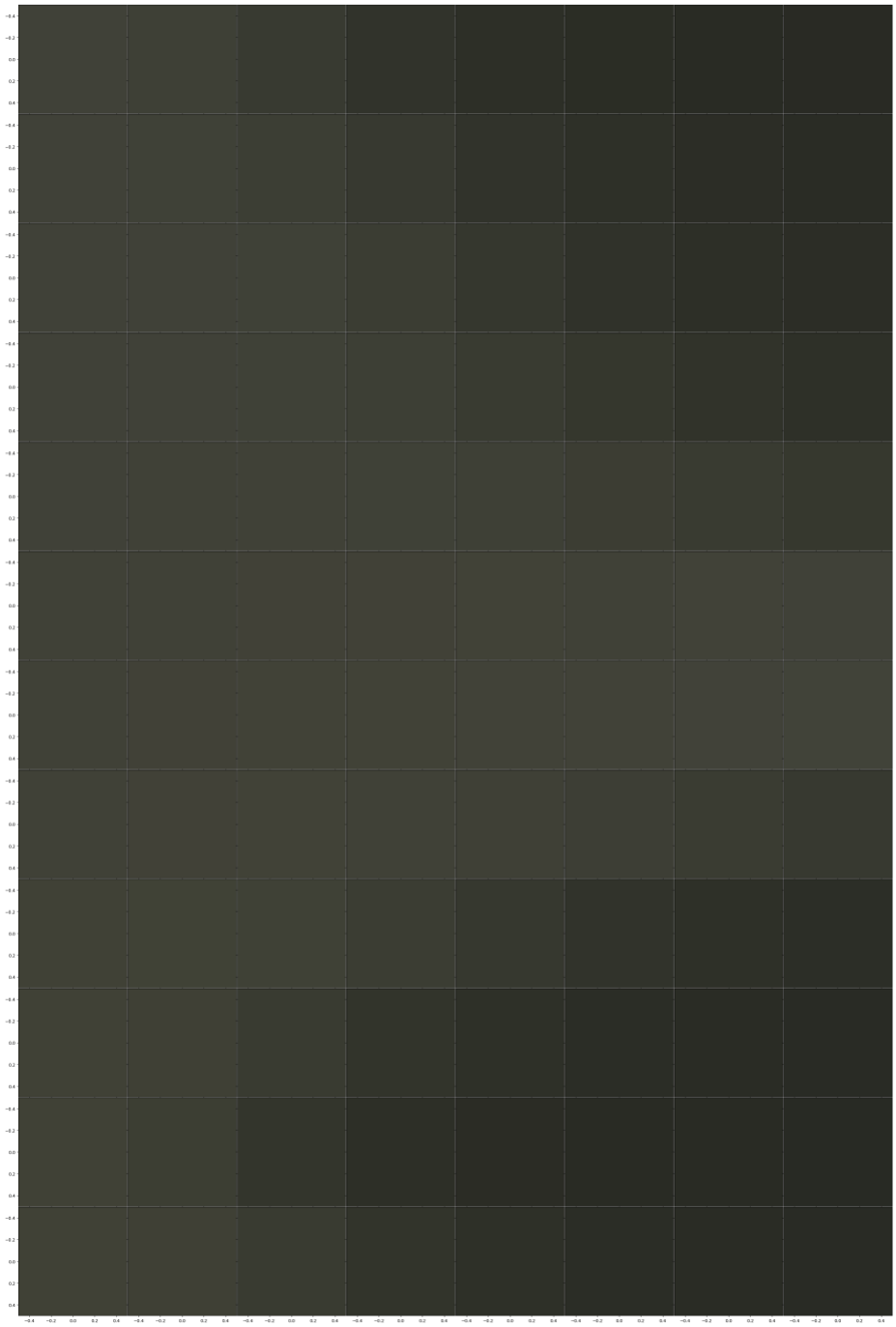
[Focal-aperture stack and confocal stereo]

1) Focal-aperture stack

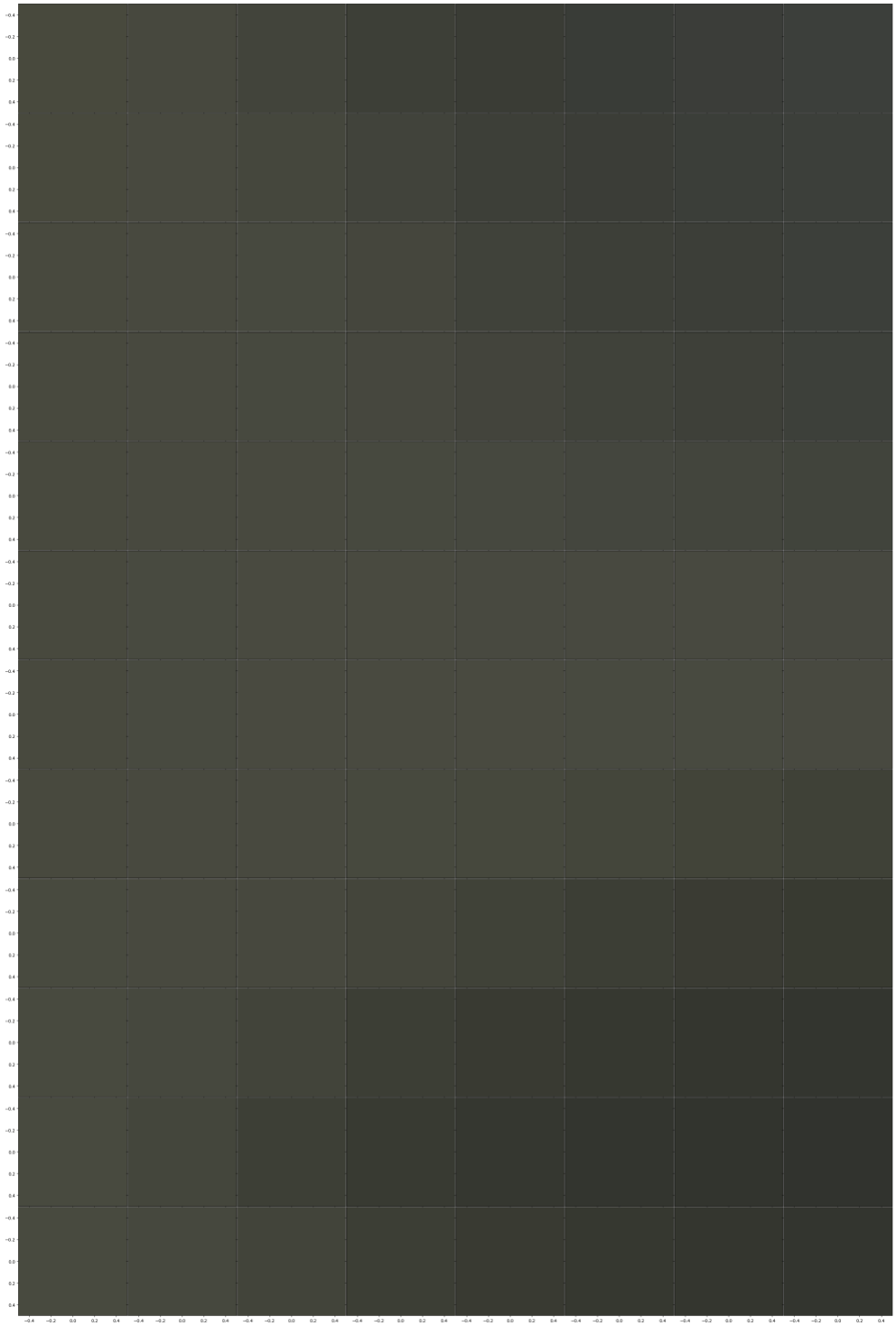

d=-0.25, a=2
d=-0.25, a=4
d=-0.25, a=6
d=-0.25, a=8
d=-0.25, a=10
d=-0.25, a=12
d=-0.25, a=14
d=-0.25, a=16
d=-0.1, a=2
d=-0.1, a=4
d=-0.1, a=6
d=-0.1, a=8
d=-0.1, a=10
d=-0.1, a=12
d=-0.1, a=14
d=-0.1, a=16
d=0, a=2
d=0, a=4
d=0, a=6
d=0, a=8
d=0, a=10
d=0, a=12
d=0, a=14
d=0, a=16
d=0.1, a=2
d=0.1, a=4
d=0.1, a=6
d=0.1, a=8
d=0.1, a=10
d=0.1, a=12
d=0.1, a=14
d=0.1, a=16
d=0.25, a=2
d=0.25, a=4
d=0.25, a=6
d=0.25, a=8
d=0.25, a=10
d=0.25, a=12
d=0.25, a=14
d=0.25, a=16
d=0.5, a=2
d=0.5, a=4
d=0.5, a=6
d=0.5, a=8
d=0.5, a=10
d=0.5, a=12
d=0.5, a=14
d=0.5, a=16
d=0.75, a=2
d=0.75, a=4
d=0.75, a=6
d=0.75, a=8
d=0.75, a=10
d=0.75, a=12
d=0.75, a=14
d=0.75, a=16
d=1, a=2
d=1, a=4
d=1, a=6
d=1, a=8
d=1, a=10
d=1, a=12
d=1, a=14
d=1, a=16
d=1.25, a=2
d=1.25, a=4
d=1.25, a=6
d=1.25, a=8
d=1.25, a=10
d=1.25, a=12
d=1.25, a=14
d=1.25, a=16
d=1.5, a=2
d=1.5, a=4
d=1.5, a=6
d=1.5, a=8
d=1.5, a=10
d=1.5, a=12
d=1.5, a=14
d=1.5, a=16
d=1.75, a=2
d=1.75, a=4
d=1.75, a=6
d=1.75, a=8
d=1.75, a=10
d=1.75, a=12
d=1.75, a=14
d=1.75, a=16
d=1.5, a=2
d=1.5, a=4
d=1.5, a=6
d=1.5, a=8
d=1.5, a=10
d=1.5, a=12
d=1.5, a=14
d=1.5, a=16

2) AFI

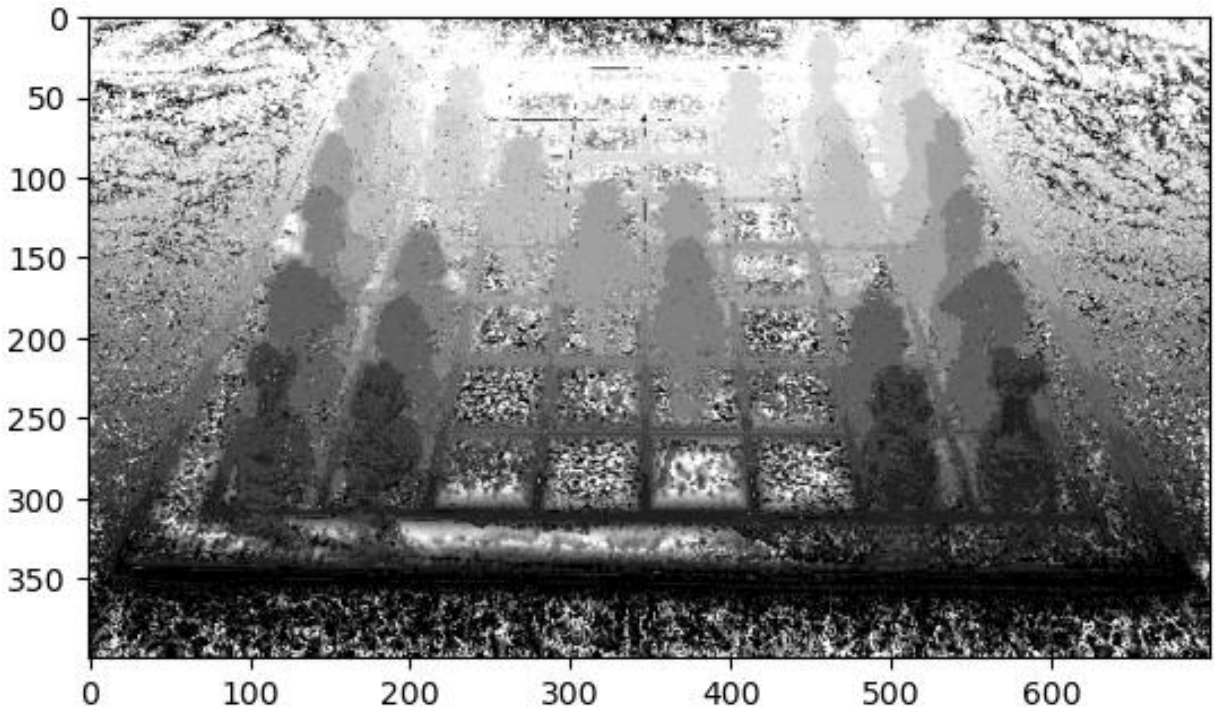
Pixel [181, 300]



Pixel [180, 490]



3) Depth map



The depth map is determined for each individual pixel, allowing it to record much more high-resolution depth information that is overlooked in the previous method. However, it tends to lack continuity. Unlike this per-pixel approach, the depth from focus stack method does not calculate on a pixel-by-pixel basis and results in a depth map that is inherently smoother and more consistent.

3. Capture and refocus your own lightfield

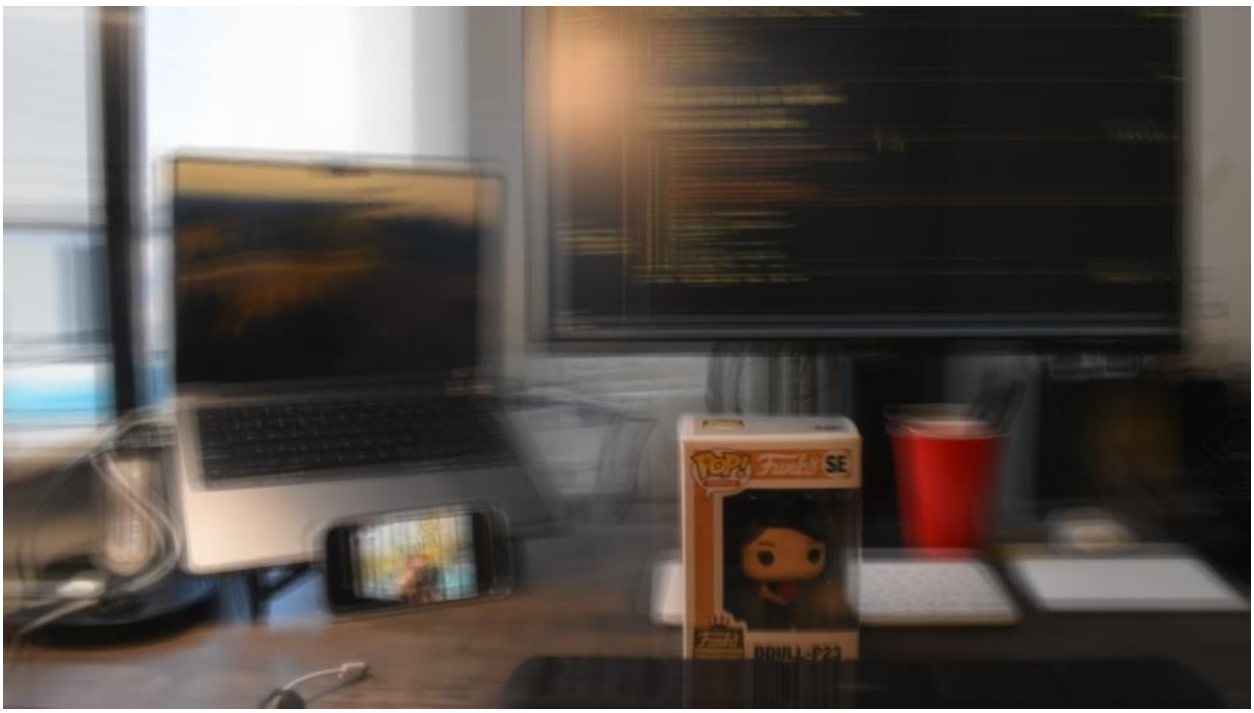
[Refocusing an unstructured lightfield]

```
def compute_norm_xcorr(img, template, bbox):
    """
    Compute normalized cross-correlation between an image and a template.

    Args:
    img (np.array): The image to search within.
    template (np.array): The template image to search for.
    bbox (tuple): A bounding box defined as (top, left, bottom, right).

    Returns:
    np.array: Normalized cross-correlation coefficients.
    """
    d1 = np.sum((template - template.mean())**2)
    norm_xcorr = 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_bar = img[i:i+template.shape[0], j:j+template.shape[1]] / img[i:i+template.shape[0],
j:j+template.shape[1]].sum()
```

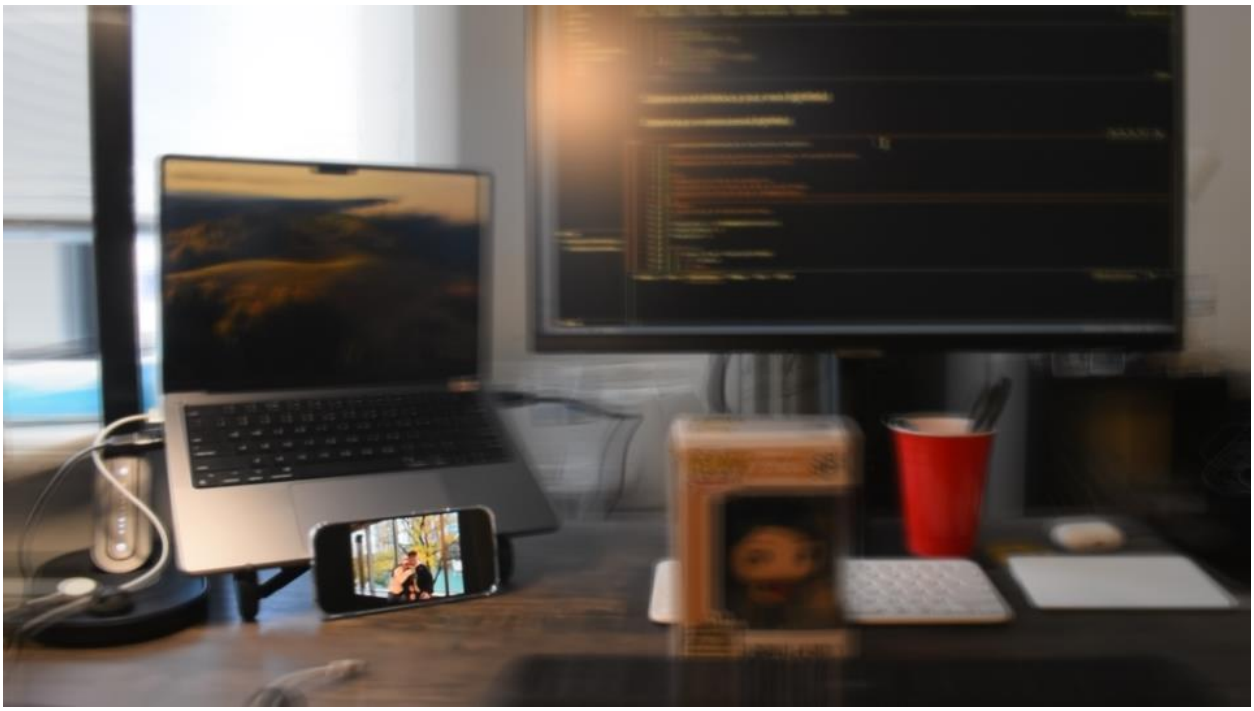
```
        d2 = np.sum((img[i:i+template.shape[0], j:j+template.shape[1]] - I_bar) ** 2)
        norm_xcorr[i,j] = num / np.sqrt(d1*d2)


return norm_xcorr
```

As shown in the above code, the equation (9) was implemented pixel-wise through the whole image.

Focused on Fanko



Focused on photo



Focused on red cup