ASSIGNMENT 4

15663 Computational Photograpy Fall 2023
DUE: November 3, 2023

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

Initials (5 points) Load the lightfield image in Python, and create from it a 5-dimensional array $L(u, v, s, t, c)$

Sub-aperture views (20 points) By rearranging the pixels in the lightfield image, we can create images that correspond to views of the scene through a pinhole placed at different points on the camera aperture. Create all of these sub-aperture views, and arrange them into a 2D mosaic.

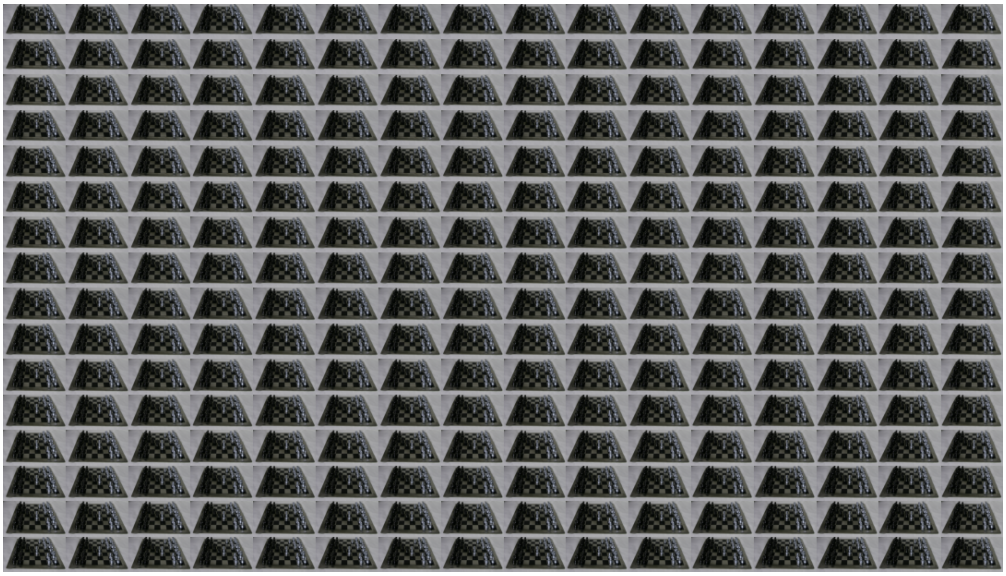

Figure 1: Mosiac of sub-aperture views

Refocusing and focal-stack simulation (25 points) Summing sub-aperture views results in an image that is focused at a depth near the top of the chess board. This corresponds to creating an image as

$$\int\int_{(u,v\in A} L(u, v, s, t, c)dvdu.$$

To focus at depth d, we must combine the sub-aperture images as:

$$I(s, t, c, d) = \int\int_{(u,v)\in A} L(u, v, s + d \cdot u, t + d \cdot v, c)dvdu.$$

The refocused images computed using the above equation:

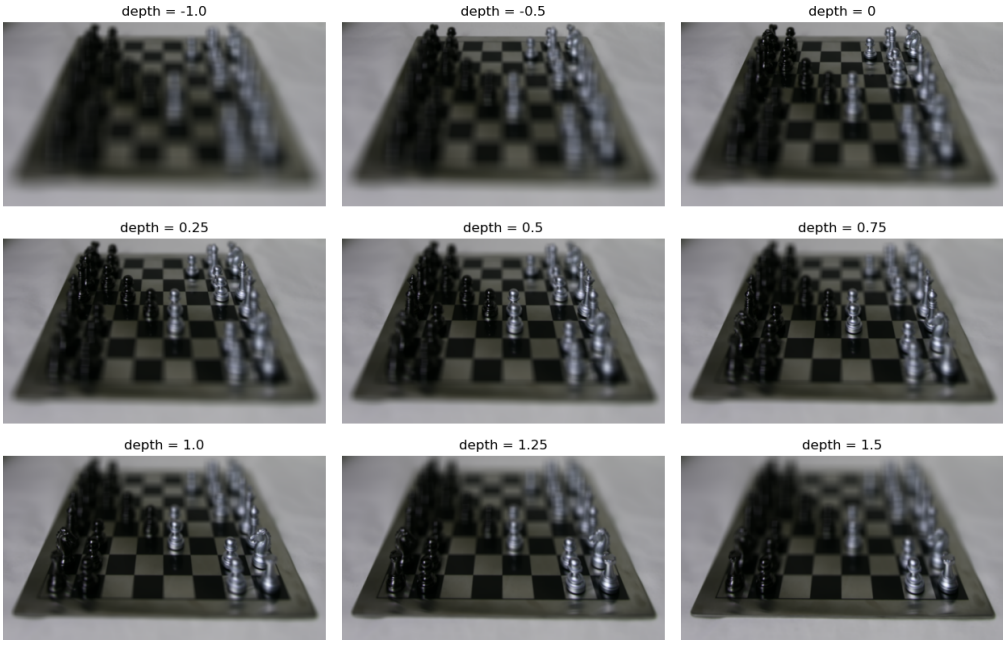

Figure 2: Refocused images

All-in-focus image and depth from focus (25 points) To merge the focal stack into a single all-in-focus image, we first need to determine per-pixel and per-image weights. In particular, the weights $w_{sharpness}(s,t,d)$ corresponds to how "sharp" the neighborhood of each pixel is at each image in the focal stack.

Once you have the sharpness weights, you can compute the all-in-focus image as:

$$I_{all-in-focus}(s,t,c) = \frac{\sum_d w_{sharpness}(s,t,d) I(s,t,c,d)}{\sum_d w_{sharpness}(s,t,d)}$$

The size of the Gaussian kernel is 21 for both kernels correspond with σ_1 and σ_2.

Experimenets with different values of σ_1 and σ_2:

$\sigma_1 = 1.0$:

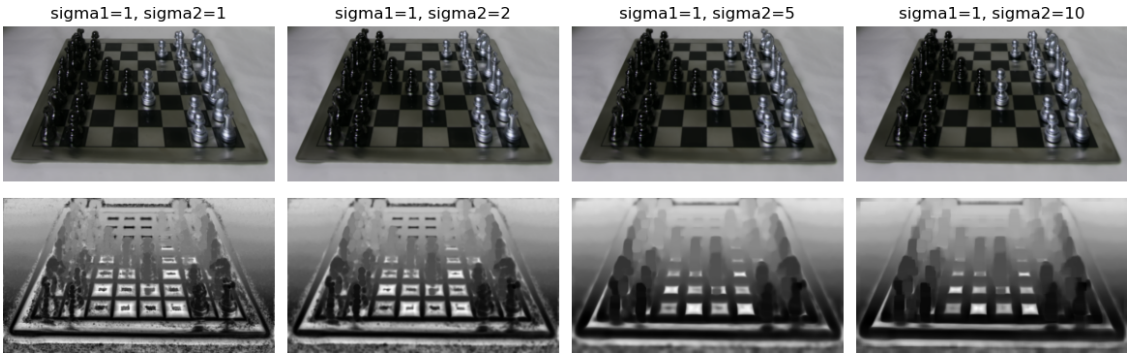

Figure 3: All focus images for $\sigma_1 = 1.0$ and different σ_2 values

$\sigma_1 = 2.0$:

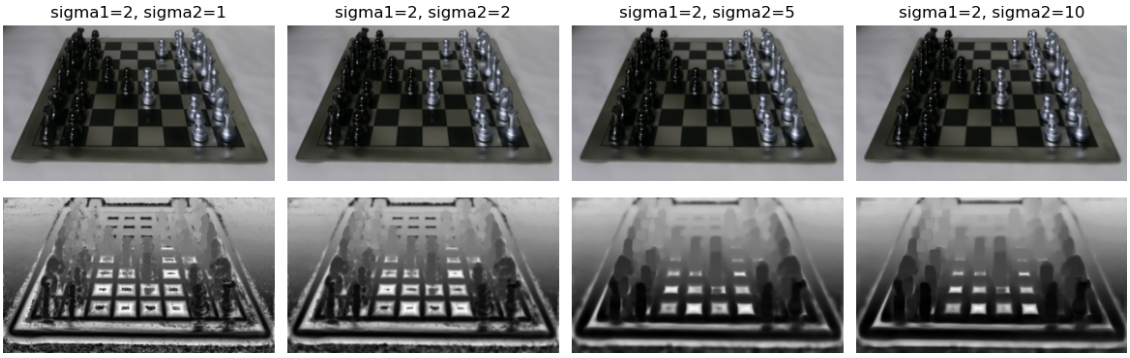

Figure 4: All focus images for $\sigma_1 = 2.0$ and different σ_2 values

$\sigma_1 = 5.0$:

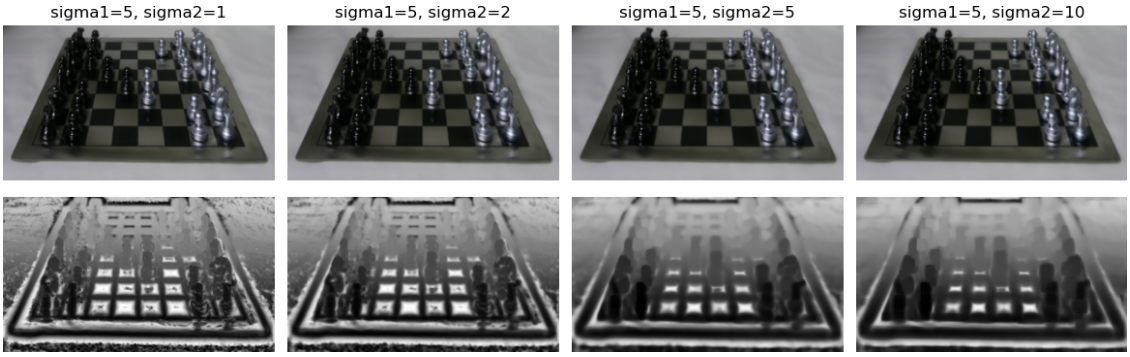

Figure 5: All focus images for $\sigma_1 = 5.0$ and different σ_2 values

$\sigma_1 = 10.0$:

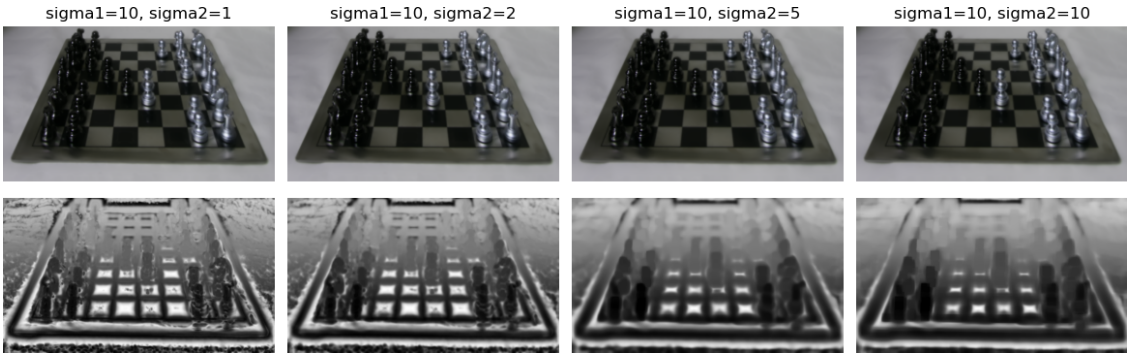

Figure 6: All focus images for $\sigma_1 = 10.0$ and different σ_2 values

The algorithm works best when we set $\sigma_1 = 2.0$ and $\sigma_2 = 10.0$. When the size of the Gaussian kernel that is used to blur the square of high-frequency component is set to 31:

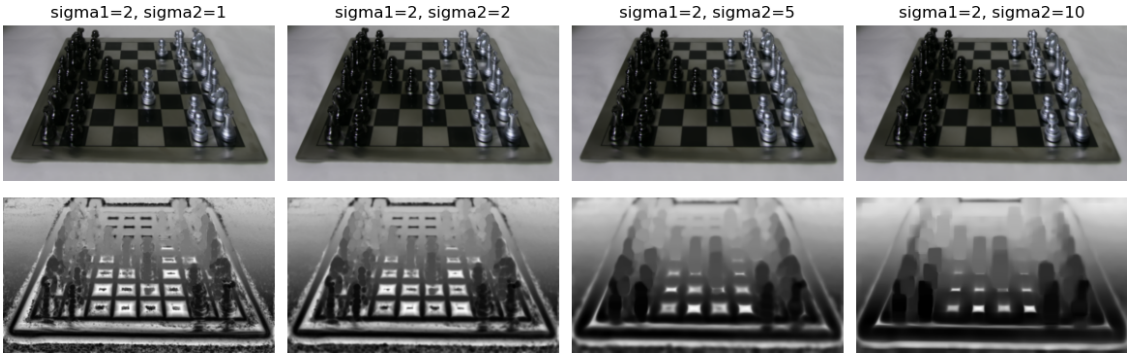

Figure 7: All focus images for $\sigma_1 = 20.0$ and different σ_2 values

The all-in-focus image and depth map computed when $\sigma_1 = 2.0$, $kernel\_size1 = 21$, $\sigma_2 = 10.0$, $kernel\_size2 = 31$:

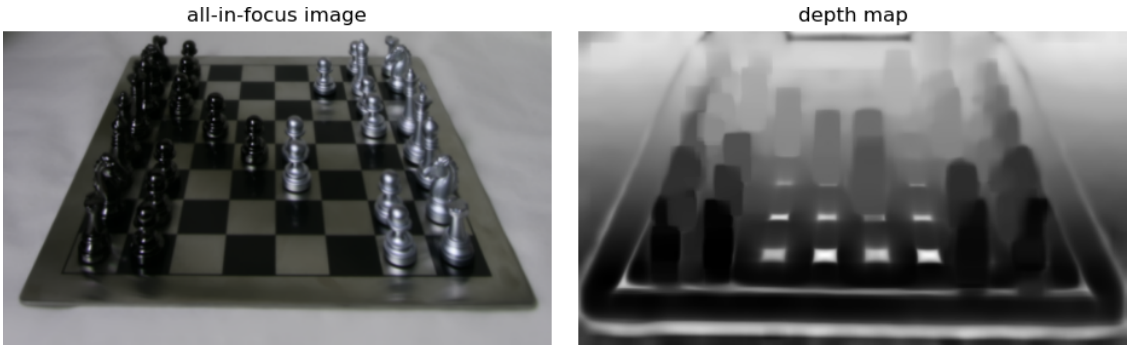

Figure 8: Left: All focus image for $\sigma_1 = 2.0$ and $\sigma_2 = 10.0$. Right: Corresponding depth map.

Changing the values of σ_1 and σ_2 does not have a visually large effect on the all-in-focus image. We use a relatively small σ_1 value which appears to balance between capturing enough details and not too much noise, and σ_2 to get a smooth result.

The grids that are closer to the camera on the checkerboard and the edges of the board have their depths incorrectly estimated. It is hard to determine the depth of these areas because they has very little texture. Because of this, for some pixels, they appear the same in all sub-aperture views, thus making it hard to infer the depth. The all-in-focus image is not quite affected at those parts because the pixel value of those parts are very close in all images in the focal stack.

Focal-aperture stack and confocal stereo (25 points) It is possible to compute *pixel-wise* depth by capturing a focal-aperture stack.

The 2D collage of the focal-aperture stack:

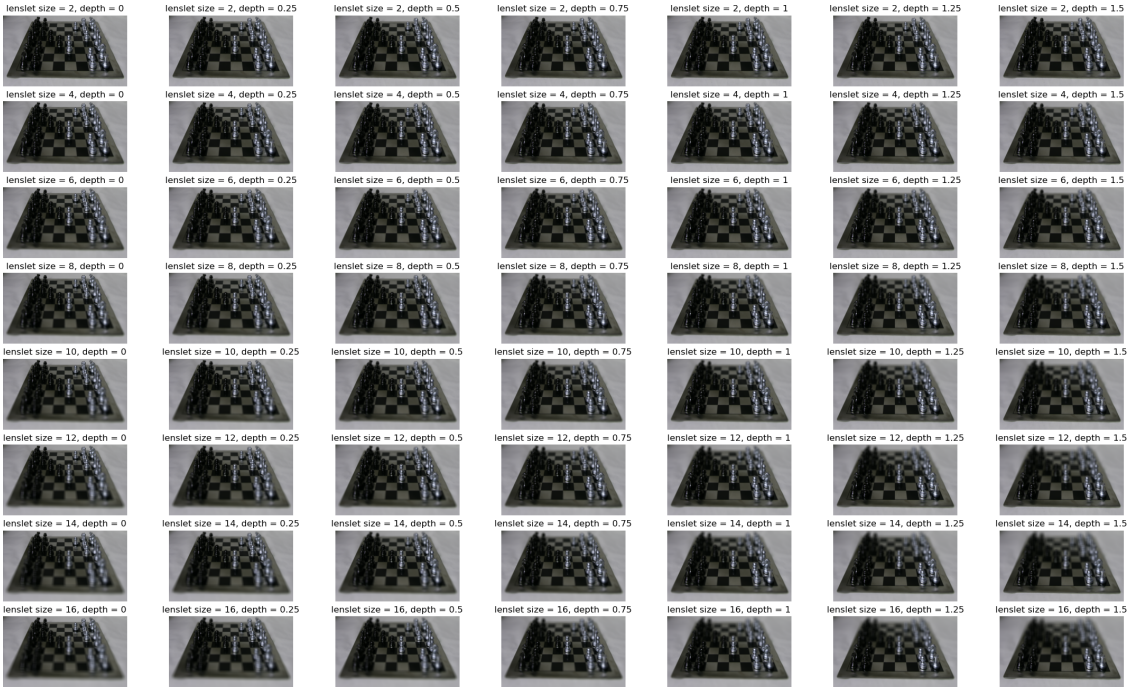

Figure 9: Focal-aperture stack

Some examples of the per-pixel aperture-focus images (AFIs):

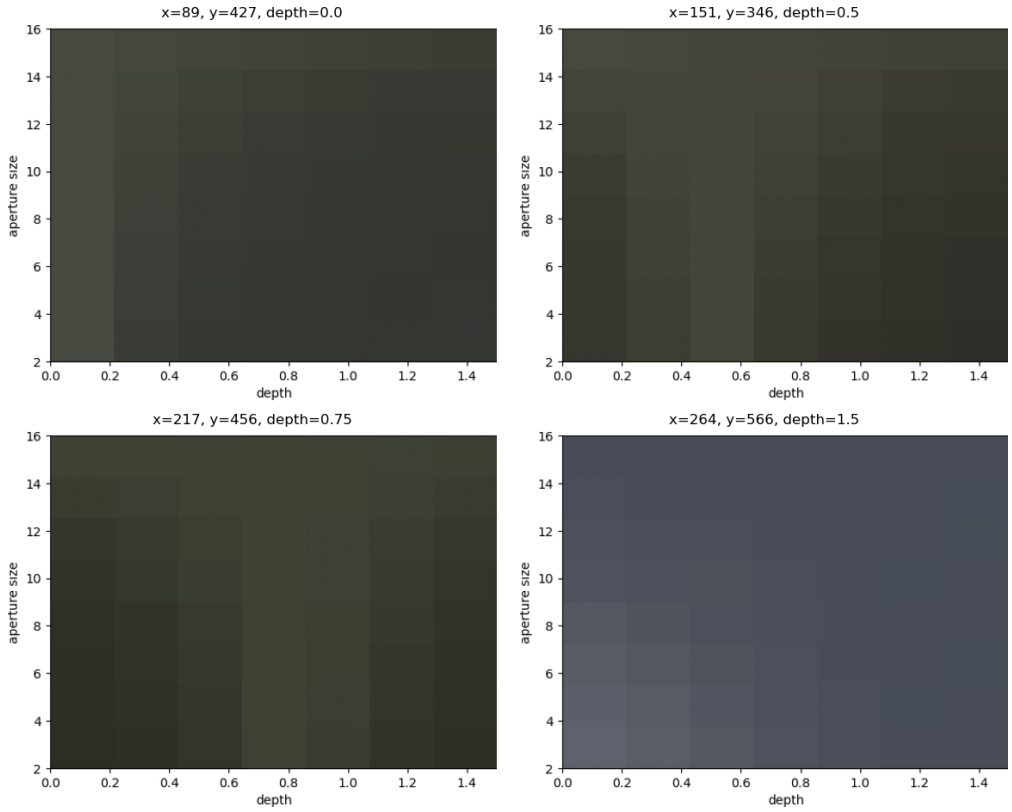

Figure 10: Examples of AFIs

The reconstructed depth map:

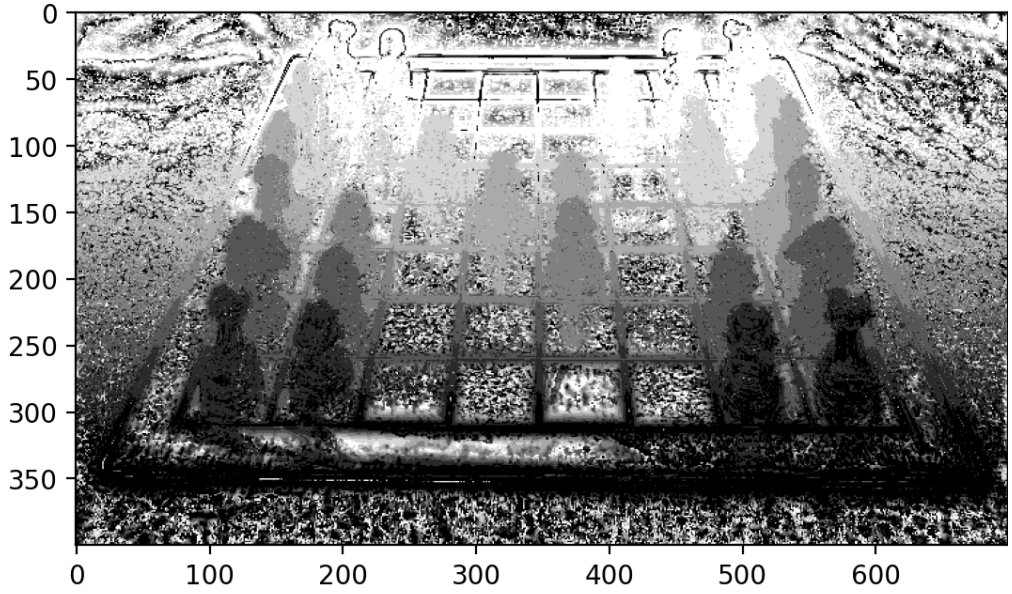

Figure 11: Reconstructed per-pixel depth map

The per-pixel depth map can capture finer details, but it is not very continuous and the depth of neighboring pixels can vary greatly. On the contrary, depth map acquired using the depth from focus procedure is smoother.

2 Capture and refocus your own lightfield (100 points)

Capturing an unstructured lightfield (30 points) We created a video of 305 frames, and use frames 150-250 for this assignment. The video is of 1920×1080, for faster process, the frames are resized to 1280×720. We also subsampled the video every second frame.

The video is under the `data` folder. Besides, it can also be viewed through this link.

Refocusing an unstructured lightfield (70 points) In the middle frame of our video sequence, select a small square neighborhood as the template around the part of the image that you want to be in focus. Then use the normalized cross-correlation method for template matching. The normalized cross-correlation equals:

$$h_t[i,j] = \frac{\sum_{k,l}(g[k,l] - \bar{g})(I_t[i+k, j+l] - \bar{I}_t[i,j]}{\sqrt{\sum_{k,l}(g[k,l] - \bar{g})^2 \sum_{k,l}(I_t[i+k, j+l] - \bar{I}_t[i,j])^2}}$$

We refocused the image on three spots: the butterfly's wing, the thermos bottle, and the hair clip. We used template size 40×40 and search window size 210×210 for the first two and template size 60×60 and search window size 250×250 for the hair clip.

We first compute the low frequency component by filtering the template with a box filter, whose size is the same as the template and its sum is normalized to 1. The result is just a single number. Then we acquire the high frequency component by subtracting this low frequency component from the template image. For every frame, within the search window, we perform template matching by finding the high

frequency component for all template-size patches and calculating the normalized cross-correlation $h_t[i, j]$, where i and j represents the coordinates of the center of the patch. The patch within the search window whose has the largest $h_t[i, j]$ is selected as the template match, and the corresponding i and j are used as the shift to shift the frame.

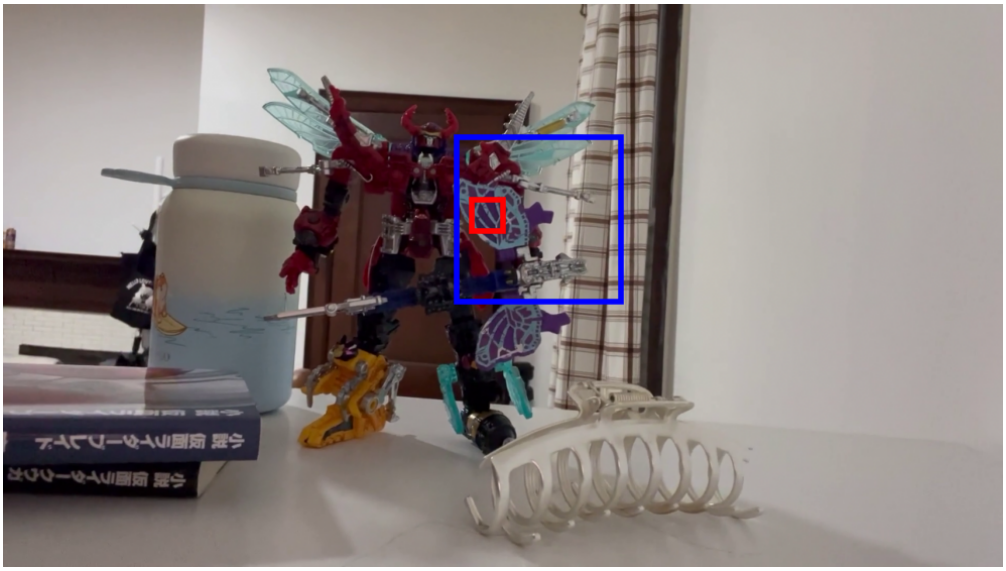

Figure 12: Template and the search window.

Frames from the unstructured lightfield and the template matches:

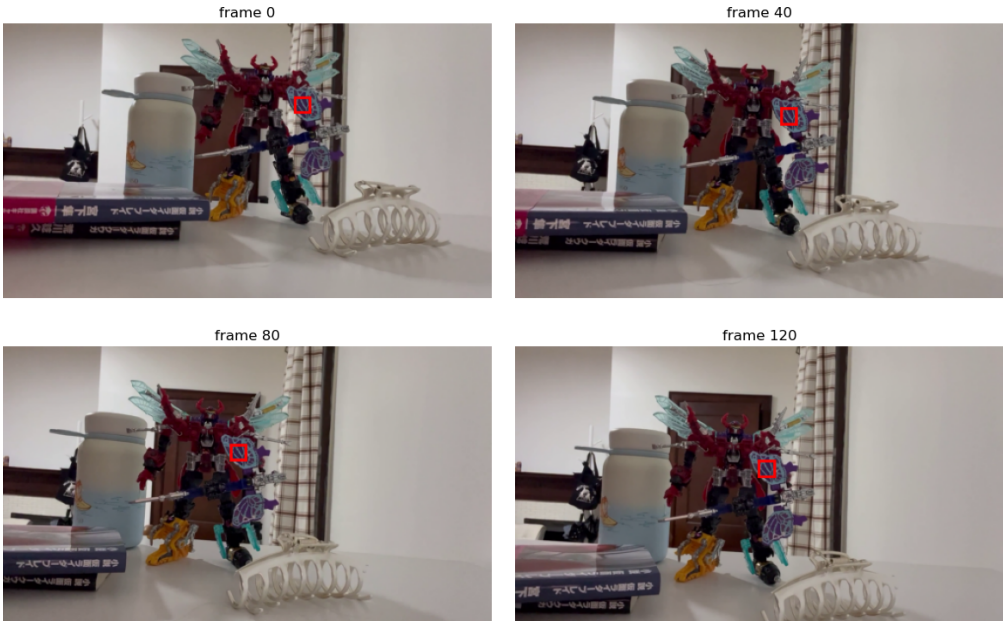

Figure 13: Frames from the unstructured lightfield

Results of focusing at different parts of the captured video:

Figure 14: Refocusing result

Refocusing results that focus at different parts of the captured video:

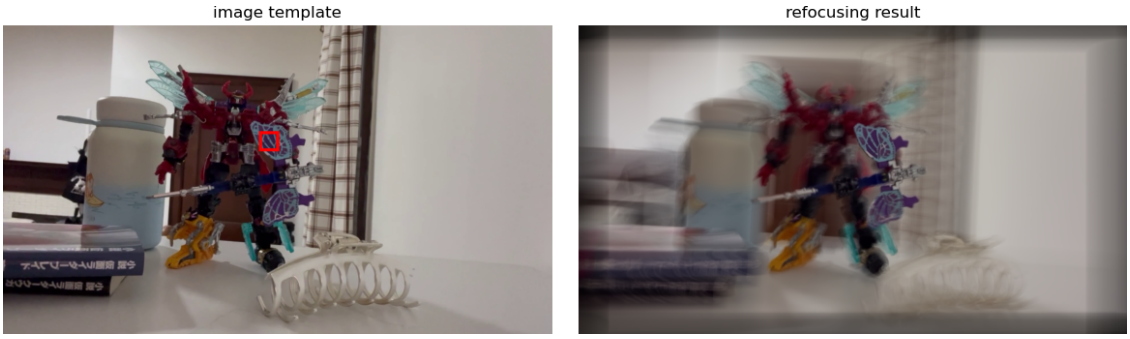

Figure 15: Left: Middle frame and the template. Right: Refocusing result.

Figure 16: Left: Middle frame and the template. Right: Refocusing result.

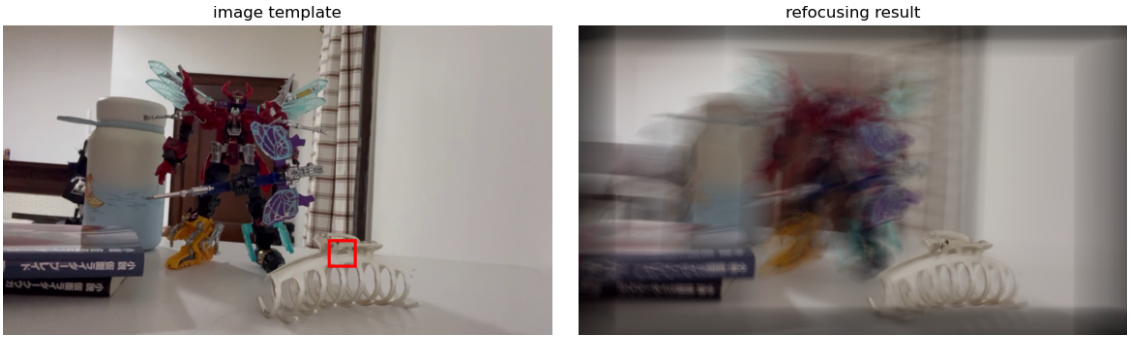

Figure 17: Left: Middle frame and the template. Right: Refocusing result.