Homework assignment 4

15-463, 15-663, 15-862 Computational Photography, Fall 2026
Carnegie Mellon University

Due Friday, Oct. 30, at 11:59pm ET

The purpose of this assignment is to explore lightfields, focal stacks, and depth from focus. As we discussed in class, having access to the full lightfield of a scene allows creating images that correspond to different viewpoint, aperture, and focus settings. We also discussed how having a focal stack of a scene allows creating an all-in-focus image and a depth map using depth from focus.

Here, you will combine the two into a single pipeline: Instead of capturing a focal stack, you will synthesize one from a lightfield image captured with a plenoptic camera. Then, from the synthetic focal stack, you will compute an all-in-focus image and a depth map.

In the first part of the homework you will be using a lightfield image captured by us. In the second part, you will capture and refocus a lightfield of your own using a standard camera.

We strongly encourage you to read the handheld plenoptic camera paper by Ng et al. [4] that we discussed in class. Sections 3.3 and 4, in particular, will be very helpful for completing this homework assignment.

Towards the end of this document, you will find a "Deliverables" section describing what you need to submit. Throughout the writeup, we also mark in red questions you should answer in your submitted report. Lastly, there is a "Hints and Information" section at the end of this document that is likely to help. We strongly recommend that you read that section in full before you start to work on the assignment.

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

For the first part of this assignment, you will use a lightfield image of a chess board scene from the Stanford Light Field Archive [1]. (We also used this scene for related examples during the lightfield lecture.) The lightfield is available as file `./data/chessboard_lightfield.png` in the homework ZIP archive. This image file is formatted in the same way as images captured by a plenoptic camera, with the pixels under neighboring lenslets corresponding to neighboring patches in the image. Ng et al. [4] describe this format in detail. Figure 1 shows a center crop of the lightfield image, and a pinhole view of the chessboard scene.

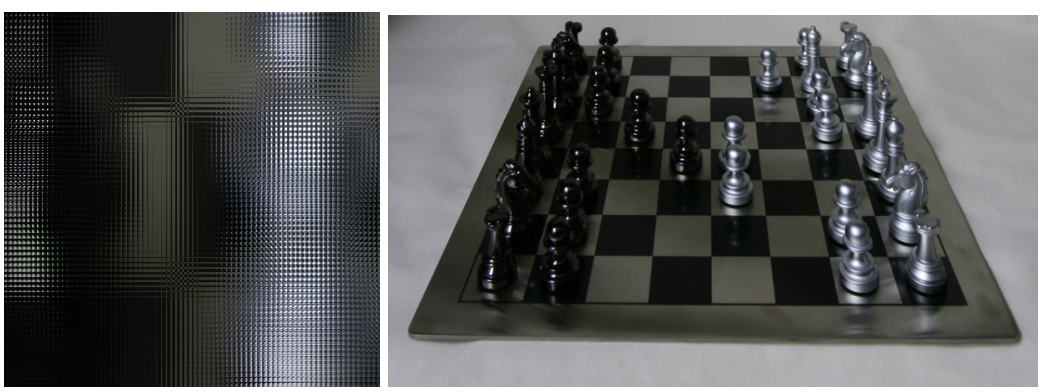

Figure 1. The chessboard lightfield. Left: Crop of the lightfield image. Right: Pinhole view of the scene.

Initials (5 points). Load the lightfield image in Python, and create from it a 5-dimensional array $L(u, v, s, t, c)$. The first four dimensions correspond to the 4-dimensional lightfield representation we discussed in class, with u and v being the coordinates on the aperture, and s and t the coordinates on the lenslet array. The fifth dimension $c = 3$ corresponds to the 3 color channels. When creating this structure, you can use the fact each lenslet covers a block of 16×16 pixels.

Sub-aperture views (20 points). As we discussed in class, 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 (a "sub-aperture"). This is equivalent to taking a slice of the lightfield of the form $L(u = u_o, v = v_o, s, t, c)$, for some values of u_o and v_o corresponding to the point on the aperture where we place the pinhole. For the chessboard lightfield, we can generate 16×16 such images.

Create all of these sub-aperture views, and arrange them into a 2D mosaic, where the vertical dimension will correspond to increasing u values, and the horizontal dimension to increasing v values. Figure 2 shows the expected result. Submit the mosaic with your solution.

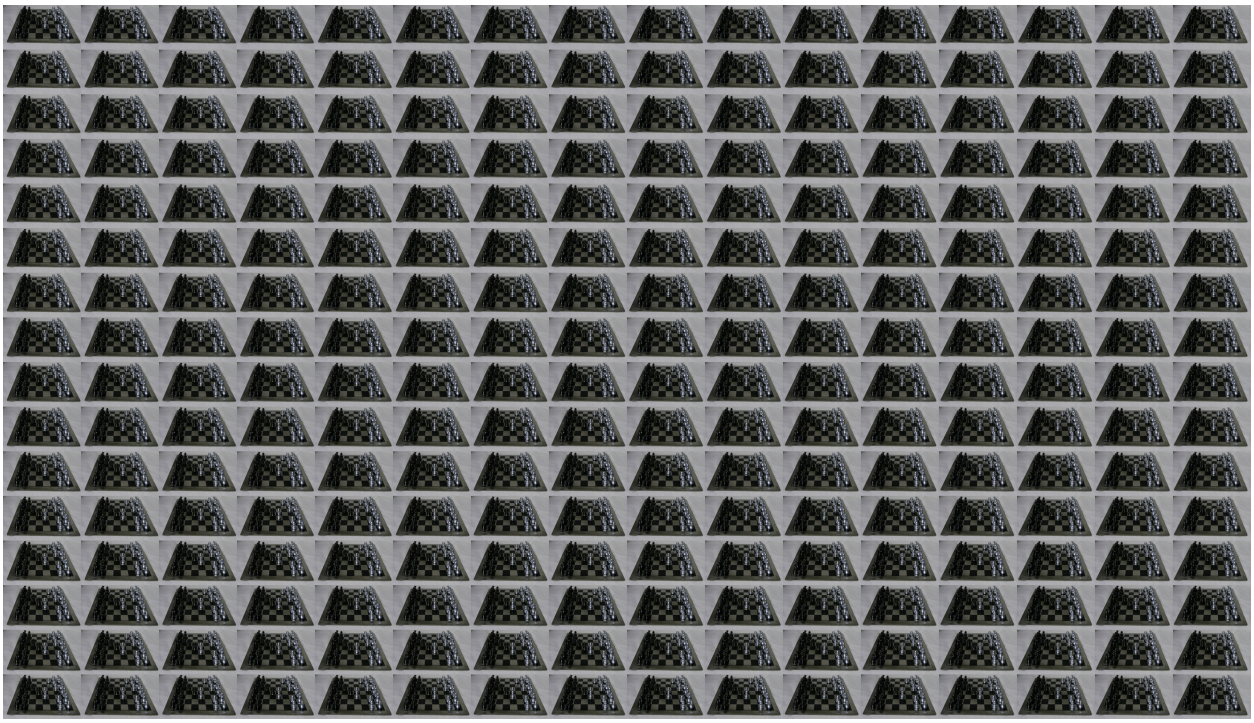

Figure 2. Mosaic of sub-aperture views.

Refocusing and focal-stack simulation (25 points). A different effect that can be achieved by appropriately combining parts of the lightfield is refocusing at different depths. In particular, 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:

$$\iint_{(u,v)\in A} L(u, v, s, t, c)\, \mathrm{d}v\, \mathrm{d}u. \tag{1}$$

The integration domain A determines the aperture size you simulate. For example, to simulate the largest aperture possible for the chess board lightfield, you should set $A = \{(u, v) : |u| \leq \text{maxUV}, |v| \leq \text{maxUV}\}$, where we express u and v relative to the center of each lenslet. To simulate a *square* aperture with side length $a < 2 \cdot \text{maxUV}$, you should set $A = \{(u, v) : |u| \leq a/2, |v| \leq a/2\}$. Lastly, to simulate a *circular* aperture of diameter $a < 2 \cdot \text{maxUV}$, you should set $A = \left\{(u, v) : \sqrt{u^2 + v^2} \leq a/2\right\}$. Note that, even though we express refocusing as a continuous integral, in practice you will be implementing a discrete sum over a finite number of (u, v) pairs, corresponding to the pixels of each lenslet. (See "Hints and Information" about maxUV and how to generate all (u, v) pairs expressed relative to the center of each lenslet.)

The left part of Figure 3 shows the resulting image using the largest aperture. As we discussed in class, and as Ng et al. [4] explain in Section 4 of their paper, focusing at different depths requires shifting then summing the sub-aperture images, with the shift of each image depending on the desired focus depth and the location of its sub-aperture. Concretely, to focus at depth d, we must combine the sub-aperture images as:

$$I(s,t,c,d) = \iint_{(u,v)\in A} L(u,v,s+d\cdot u,t+d\cdot v,c)\,\mathrm{d}v\,\mathrm{d}u. \tag{2}$$

In the above equation, note the difference between the italicized d, which stands for focusing depth, and the upright d, which is a differential. For $d=0$, the image we obtain is the same as in Equation (1). Figure 3 shows refocused images for two more settings of d, again using the maximum aperture.

Figure 3. Refocusing at different depths using the largest aperture. The left image corresponds to using Equation (2) with $d=0$ (or equivalently, using Equation (1)). These images have had their exposure adjusted to match that of the images in Figure 2, by dividing the result of Equation (2) by the size of the aperture.

Implement Equation (2) using the largest aperture, and use it to simulate a focal stack $I(s,t,c,d)$ for a range of values d. Make sure that your depth range is long enough so that each part of the scene is in focus in at least one image in the stack. In your solution, make sure to show at least five different refocusings.

All-in-focus image and depth from focus (25 points). As we saw in class, when we have access to a focal stack, we can merge the images into a new image where all the scene is in focus. In the process of doing so, we also obtain depth estimates for each part of the scene, a procedure known as *depth from focus*.

To merge the focal stack into a single all-in-focus image, we first need to determine per-pixel and per-image weights. This is similar to the procedure used in homework assignment 2 for high-dynamic range imaging, with the difference that the weights here are very different. In particular, the weights in this case correspond to how "sharp" the neighborhood of each pixel is at each image in the focal stack.

There are many possible sharpness weights. Here you will implement the following:

1. For every image in the focal stack, first convert it to the XYZ colorspace (making sure to account for gamma encoding), and extract the luminance channel:

$$I_{\text{luminance}}(s,t,d) = \text{get_luminance}(\text{rgb2xyz}(I(s,t,c,d))). \tag{3}$$

2. Create a low-frequency component by blurring it with a Gaussian kernel of standard deviation σ_1:

$$I_{\text{low-freq}}(s,t,d) = G_{\sigma_1}(s,t) * I_{\text{luminance}}(s,t,d). \tag{4}$$

3. Compute a high-frequency component by subtracting the blurry image from the original:

$$I_{\text{high-freq}}(s,t,d) = I_{\text{luminance}}(s,t,d) - I_{\text{low-freq}}(s,t,d). \tag{5}$$

4. Compute the sharpness weight by blurring the *square* of high-frequency component with another Gaussian kernel of standard deviation σ_2:

$$w_{\text{sharpness}}(s,t,d) = G_{\sigma_2}(s,t) * (I_{\text{high-freq}}(s,t,d))^2. \tag{6}$$

The weights are the same for each of the color channels.

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

$$I_{\text{all-in-focus}}(s,t,c) = \frac{\sum_d w_{\text{sharpness}}(s,t,d) I(s,t,c,d)}{\sum_d w_{\text{sharpness}}(s,t,d)}. \tag{7}$$

You can also create a depth map by using the weights to merge depth values instead of pixel intensities:

$$\text{Depth}(s,t) = \frac{\sum_d w_{\text{sharpness}}(s,t,d) d}{\sum_d w_{\text{sharpness}}(s,t,d)}. \tag{8}$$

Figure 4 shows the all-in-focus image and depth map resulting from one set of σ_1 and σ_2 values used for sharpness evaluation. You should experiment with different values and report which ones work best, as well as show the corresponding all-in-focus image and depth map. Which parts of the depth map have their depth incorrectly estimated, and why? Is the all-in-focus image similarly affected at those parts, and why?

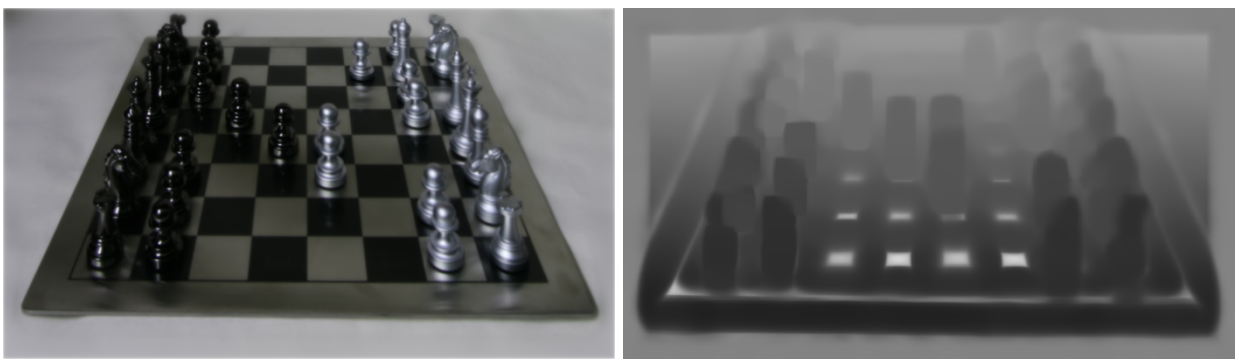

Figure 4. Left: All focus image for one set of σ_1 and σ_2 values. Right: Corresponding depth map.

Focal-aperture stack and confocal stereo (25 points). As we discussed in class, Hassinoff and Kutulakos [3] show that it is possible to compute *pixel-wise* depth by capturing a focal-aperture stack. (There is a subsequent journal version of the confocal stereo paper, published at IJCV. The section and equation references in this writeup are with respect to the ECCV version of the paper cited in the bibliography.)

Implement Equation (2) for different aperture sizes, and use it to simulate a focus-aperture stack. You can choose whether to simulate square or circular apertures. Then, estimate pixel-wise depth from your stack using the procedure Hassinoff and Kutulakos [3] describe in Equations (7) and (8) of their paper. In Equation (7), you can set the exitance E_{xy} for each aperture to be equal to the number of (u,v) pairs included in the integration domain A in Equation (2). Note that, if you have already been dividing the result of Equation (2) by the aperture size in previous parts, to adjust image exposure, then you should *not* divide a second time here. Note also that you do *not* need to implement the alignment procedure the paper describes.

Make sure to show a 2D collage of your focal-aperture stack, a few example per-pixel aperture-focus images (AFIs), and the reconstructed depth map. How does the depth map you recover here compare to the one from the depth-of-focus procedure you implemented earlier?

2. Bonus: Better blending and depth map (20 points)

In the past several lectures, we have discussed a variety of techniques for computing "sharpness" and blending images. In this bonus question, you can experiment with different techniques for blending a focal stack into an all-in-focus image, as well as extracting a depth map for it.

This bonus part is intentionally open-ended, to give you an opportunity to experiment and explore. You are also welcome to look into related literature for ideas—the references at the end of the lecture on focal

stacks and lightfields should be a good starting point. How many points you get for this bonus part will depend on three factors: 1) the magnitude of the experiments you perform (e.g., just replacing the weighting method of Part 1 with running a Laplacian will not get you many points); 2) the novelty and soundness of the blending pipeline you come up with; and 3) the improvement in the all-in-focus image and depth map.

3. Capture and refocus your own lightfield (100 points)

You will now capture and refocus your own lightfield. For this, you can use either the Nikon D3300 camera you borrowed at the start of the class, or your own cell phone camera.

Capturing an unstructured lightfield. (30 points) As we saw in class, in the absence of a plenoptic camera and camera array, an easy way to capture a lightfield is to move a camera around and capture multiple images. Ideally, we would move the camera at constant x-y intervals, to create a regular sampling of the true underlying lightfield. However, this is hard to do in two dimensions without specialized equipment.

Instead, here you will capture an *unstructured lightfield* [2]. The procedure for doing this is shown in Figure 5: Use a camera to capture *video*, while moving the camera along a plane. Doing so corresponds to sampling the aperture plane in an *unstructured* way, at irregular values (u, v) that depend on the trajectory of the camera, instead of the structured grid sampling performed by a camera moving at regular x-y intervals. In your homework submission, make sure to include the video you end up using.

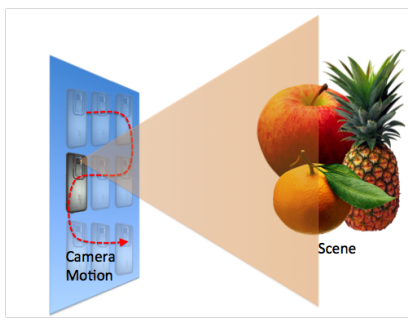

Figure 5. Capturing an unstructured lightfield.

Refocusing an unstructured lightfield. (70 points) In the first part of the homework, the fact that we knew the amount of shift corresponding to each viewpoint in the lightfield greatly simplified refocusing: We could refocus by aligning images based on the part of the scene we wanted to be in focus, with this alignment corresponding to shifts determined by the rectangular grid structure we used to sample the lightfield.

In the unstructured lightfield you captured, you do not know the sampling grid structure. Therefore, you will need to infer the amount of shift that needs to be applied for images to be aligned at a specific point.

To see how to do this, let's first look at Figure 6, which shows a few frames from an unstructured lightfield. Let's say that we want to create an image that is focused on the bug eye marked with a red box. We will determine how to shift images using a template matching procedure.

In particular, in the middle frame of your video sequence, select a small square neighborhood around the part of the image that you want to be in focus. Then, use the corresponding image patch as a template, with which you will perform template matching in all other video frames. Use the normalized cross-correlation method for template matching: Let your template be $g[i, j]$ and a video frame be $I_t[i, j]$, where t is used to index video frames. Then the normalized cross-correlation equals:

$$h_t[i,j] = \frac{\sum_{k,l}(g[k,l]-\bar{g})\big(I_t[i+k,j+l]-\bar{I}_t[i,j]\big)}{\sqrt{\sum_{k,l}(g[k,l]-\bar{g})^2 \sum_{k,l}\big(I_t[i+k,j+l]-\bar{I}_t[i,j]\big)^2}}, \tag{9}$$

Figure 6. Frames from an unstructured lightfield.

where $\bar{I}_t[i,j]$ is a version of $I_t[i,j]$ filtered with a box filter that has the same size as the template $g[i,j]$ and is normalized to sum to 1. The left part of Figure 7 shows a visualization. You must implement Equation (9) using only pixel-wise and convolution (or correlation) operations, the latter using, e.g., `scipy`'s `correlate2d`. In your report, make sure to explain exactly how you did this. Template matching should be performed using a grayscale template and on grayscale video frames (you can use the luminance channel for this).

Once you have computed the cross-correlation, for each frame of the video you can compute a shift as:

$$[s_{t,x}, s_{t,y}] = \operatorname*{argmax}_{i,j} h_t[i,j]. \tag{10}$$

Shift each frame by its corresponding amount $[s_{t,x}, s_{t,y}]$, then sum all frames. The result should be an image that is focused around the template you used. The right part of Figure 7 shows the focusing result.

Figure 7. Focusing by template matching. Left: Template matching procedure for one frame. Right: Refocusing result.

Implement this procedure, and show a few results of focusing at different parts of your captured video.

4. Bonus: Capture and process your own focal-aperture stack (50 points)

As we discussed in class, you do not need a lightfield camera to do confocal stereo, depth from focus, or create all-in-focus images. Instead, you can do the same by using your regular camera to capture a focal-aperture stack, simply by taking photographs at a sequence of different focus settings.

Use a camera with manual settings (e.g., the Nikon D3300 you borrowed for the class) to capture a dense focal-aperture stack, then process it using your confocal stereo implementation, to create a depth map. Additionally, process the focal stack images corresponding to the largest aperture using your depth from focus implementation, to create another depth map and an all-in-focus image. You should make sure to capture a scene where there is significant depth variation (see the example focal stack in the lecture slides). Additionally, you should make sure your measurements include the smallest and largest aperture available by your lens, as their corresponding large and shallow depths of field will help improve the quality of the two depth maps you produce. We strongly recommend that you capture your focal-aperture stack by tethering your camera to your laptop, and using `gphoto2` or some other software to control it remotely.

Depending on the depth range captured in your focal-aperture stack, you may need to first perform an alignment step, to account for the change in magnification as the focus changes. If this turns out to be necessary, we recommend that you perform this alignment using a simple global scaling, as discussed in class. You can use the Gaussian lens formula and the focal length and focusing distance settings reported by the camera to figure out the exact scaling you need to use.

Deliverables

When submitting your solution, make sure to follow the homework submission guidelines available on the course website (http://graphics.cs.cmu.edu/courses/15-463/assignments/submission_guidelines.pdf). Your submitted solution should include the following:

- A PDF report explaining what you did for each problem, including answers to all questions asked throughout Problems 1 and 3, as well as any of the bonus problems you choose to do. The report should include any figures and intermediate results that you think may help. Make sure to include explanations of any issues that you may have run into that prevented you from fully solving the assignment, as this will help us determine partial credit. The report should also explain any additional image files you include in your solution (see below).
- All of your Python code, including code for the bonus problems, as well as a `README` file explaining how to use the code.
- The viewpoint collage, refocused images, focal-aperture collage, AFIs, depth maps, and all-in-focus images that you create for problems 1 and 3, as well as the video you use for problem 3. You can also include additional image files for various experiments other than your final ones, if you think they show something important.
- If you do Bonus Part 2: Your PDF report should include a detailed description of the experiments you performed, including descriptions of the blending and depth-from-focus pipelines you ended up implementing. You should also show and compare the all-in-focus images and depth maps you generated with the different methods.
- If you do Bonus Part 4: Your PDF report should include a detailed description of how you captured your focal stack, as well as of how you performed alignment. Additionally, you should show example frames of the focal stack you captured, and the resulting all-in-focus image and depth map.

Hints and Information

Shifting. To implement the shifts described in various parts of the homework, you can use the `scipy` function `RegularGridInterpolator` that you are already familiar with from homework assignment 1.

Coordinate centering. When refocusing in problem 1, it will be helpful to express the coordinates u and v relative to the center of each lenslet. You can do this as follows:

```
lensletSize = 16
maxUV = (lensletSize - 1) / 2
u = np.arange(lensletSize) - maxUV
v = np.arange(lensletSize) - maxUV
```

Capturing your unstructured lightfield. The quality of your results in part 3 will critically depend on how you moved the camera during video capture. In particular, you must avoid tilting and rotating the camera as much as possible. You should just shift the camera in a plane, as in Figure 5.

When capturing the unstructured lightfield, it will likely take many attempts before you capture a video that is planar enough. We recommend first practicing the planar motion of the camera, then capturing several videos and processing them as explained in the next part, until you get a satisfactory result. Below we also list a few tips that can help with this part.

To improve the planarity of the camera motion, you can press your camera against a flat object (e.g., a book) to help guide its motion. This is more easily done with a phone camera than a DSLR.

You can experiment with different types of camera motion, but we recommend using a zig-zag motion as in Figure 5. In general, the larger part of the plane you cover, the better your refocusing results will be.

Make sure your video is not too long. You should capture a few seconds of video. Otherwise, you will need to process an enormous amount of data.

Capture a scene with a few objects at different depths. See Figure 6 for an example. This will make the visual effect of refocusing at different objects more compelling.

Make sure that all the objects in the scene are in focus. Remember that each image in a lightfield is meant to be a pinhole view of the scene, and therefore should all be in-focus.

It is also important that your camera's settings (focus, zoom, aperture, ISO, and shutter speed) remain constant throughout the video capture. If you use the DSLR camera, this is easily achievable by setting the camera to manual mode. If you use your phone camera, we recommend that you use an app that allows manual control of the camera settings (e.g., Adobe Lightroom).

Efficient template matching. The template matching-based refocusing in problem 3 is a very computationally intensive operation. Below are a few tips for making this faster.

You can use all the frames in the video, but this may make your processing very slow if you captured more than a few seconds of video. You should be able to get away with subsampling your video by some step size (e.g., every second or third frame, depending on how long your sequence is).

The size of the template you use for template matching is very important for the quality and speed of template-matching-based refocusing. A very small template will result in faster processing, but also in potentially inaccurate shifts. Conversely, a very large template will result in slower processing, but in more accurate shifts. You can experiment with template sizes between 10×10 and 100×100 pixels.

As shown in the left part of Figure 7, you only need to search for a template match within a window of your target frame. The window should be centered on the location of the template in the middle frame (where you selected the template). The size of the window should be slightly larger than the sum of the template size and the maximum shift of the target object over the entire set of video frames. For instance, in Figure 7, the template was 60×60 pixels and the maximum shift of the template was about 40 pixels, therefore the search window was 200×200 pixels.

Focal stacking. To capture and process your own focal stack, you will need to use `gphoto2` to control the focus settings of your camera. There are some hints at this this website for how to do this, but the exact steps will be affected by what camera you use. The following instructions have been tested on a Nikon D3400.

First, you will want to set the camera's viewfinder focus mode to AF-S. You should do this on the camera directly instead of trying to set the configuration through `gphoto2`. Then you will need to enter `gphoto2`'s shell mode with the command `gphoto2 --shell`.

```
#enter viewfinder mode to enable focus control
set-config /main/actions/viewfinder=1
#use large focus step to hit edge of focus range
#will produce error "Nikon manual focus stepping too small" upon hitting edge
set-config /main/actions/manualfocusdrive=1000
#alternate captures and small focus steps with the opposite sign
#no for-loop in shell-mode, so manually repeat these as necessary
capture-image-and-download
set-config /main/actions/manualfocusdrive=-150
```

Additionally, you will need to know the focusing distance of the RAW images you capture. You can use ExifTool to extract this information from the RAW file's metadata.

Credits

A lot of inspiration for this assignment came from similar assignments in computational photography courses offered by Fredo Durand at MIT, Gordon Wetzstein at Stanford, and Ollie Cossairt at Northwestern. The figures and write-up of Problem 3 in particular are mostly taken from Ollie Cossairt's course.

References

[1] The (new) stanford light field archive, 2008. http://lightfield.stanford.edu/.

[2] A. Davis, M. Levoy, and F. Durand. Unstructured light fields. In *Computer Graphics Forum*, volume 31, pages 305–314. Wiley Online Library, 2012.

[3] S. W. Hasinoff and K. N. Kutulakos. Confocal stereo. In *European Conference on Computer Vision*, pages 620–634. Springer, 2006.

[4] R. Ng, M. Levoy, M. Brédif, G. Duval, M. Horowitz, and P. Hanrahan. Light field photography with a hand-held plenoptic camera. *Stanford Computer Science Technical Report*, 2005.