Homework Assignment 4

15-463/663/862, Computational Photography, Fall 2018
Carnegie Mellon University

Due Friday, Oct. 26, at 11:59pm

The purpose of this assignment is to explore lightfields, focal stacks, and depth from defocus. 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 defocus.

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 lightfied of your own using a standard camera.

You are strongly encouraged to read the handheld plenoptic camera paper by Ng et al. [3] that we discussed in class. Sections 3.3 and 4, in particular, will be very helpful for understanding how to solve this homework assignment. Additionally, as always, there is a "Hints and Information" section at the end of this document that is likely to help.

1. Lightfield rendering, focal stacks, and depth from defocus (80 points)

For the first part of the homework, you will use a lightfield image of a chess board scene, obtained 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. This format is described in detail in [3]. Figure 1 shows a crop from the center of the lightfield image, as well as a regular pinhole-camera view of the chessboard scene.

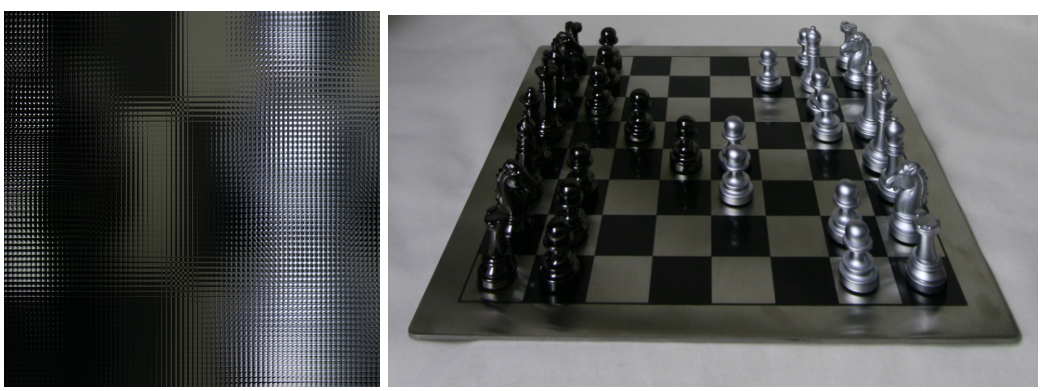

Figure 1: The chessboard scene lightfield. Left: Crop of the lightfield image. Right: A pinhole camera view of the scene.

Initials (5 points). Load the lightfield image in Matlab, and create from it a 5-dimensional array $L(u, v, s, t, c)$. The first four dimensions correspond to the 4-dimensional lightield 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 (15 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.

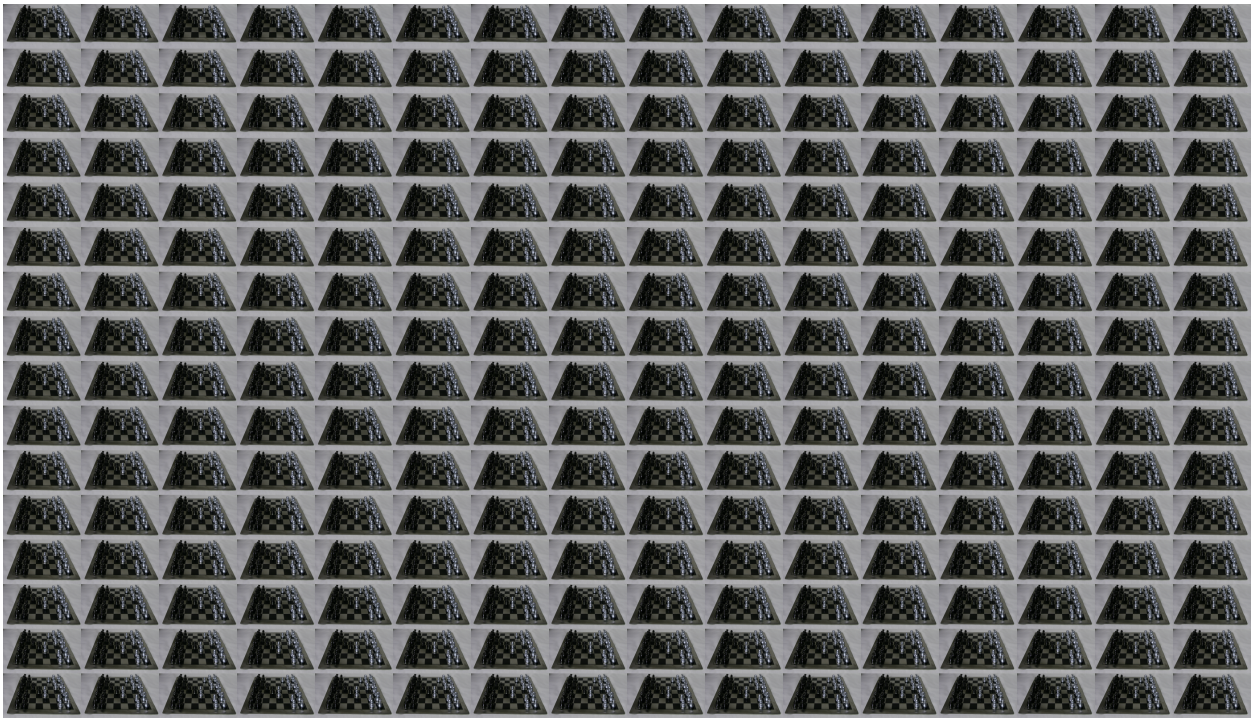

Figure 2: Mosaic of sub-aperture views.

Refocusing and focal-stack generation (30 points). A different effect that can be achieved by appropriately combining parts of the lightfield is refocusing at different depths. In particular, averaging all of the 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:

$$\int_u \int_v L(u, v, s, t, c) \, \mathrm{d}v \, \mathrm{d}u. \tag{1}$$

The resulting image is shown to the left of Figure 3. As we discussed in class and explained in detail in Section 4 of [3], focusing at different depths requires shifting the sub-aperture images before averaging them, with the shift of each image depending on the desired focus depth and the location of its sub-aperture. More concretely, to focus at depth d, we need to combine the sub-aperture images as:

$$I(s, t, c, d) = \int_u \int_v L(u, v, s + du, t + dv, c) \, \mathrm{d}v \, \mathrm{d}u. \tag{2}$$

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.

Implement Equation (2), and use it to generate a focal stack $I(s, t, c, d)$ for a range of values d. Make sure that your focal stack 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.

Figure 3: Refocusing at different depths. The left image corresponds to using Equation (2) with $d = 0$ (or equivalently, using Equation (1)).

All-focus image and depth from defocus (30 points). As we saw in class, when we have access to a focal stack, we can merge the images into a new images where all of the scene is in focus. In the process of doing so, we also obtain depth estimates for each part of the scene, a procedure that is known as *depth from defocus*.

To merge the focal stack into a single all-focus image, we first need to determine per-pixel and per-image weights. This is similar to the procedure used in Homework 4 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 tonemapping), and extract the luminance channel:

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

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:

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

Note that the weights are the same for each of the color channels.

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

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

In addition, you can create a per-pixel depth map by using the weights to merge depth values instead of pixel intensities, that is:

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

Figure 4 shows the all-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-focus image and depth map.

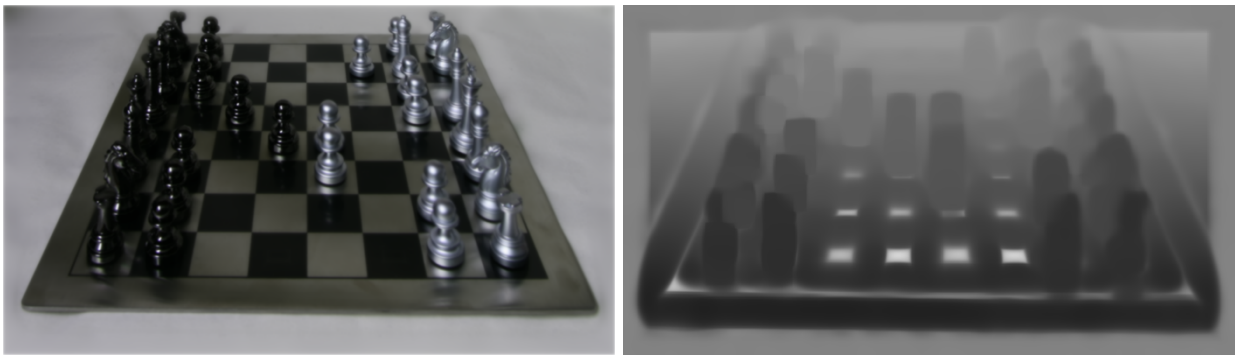

Figure 4: Left: All focus image for one set of σ_1 and σ_2 values. Right: Corresponding depth map.

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 left 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 will be awarded 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 resulting all-in-focus image and depth map.

4. Capture and refocus your own lightfield (70 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. (20 points) As we saw in class, in the absence of a plenoptic camera and a camera array, an easy way to capture a lightfield is to use a camera that we move around to capture multiple images. Ideally, we would move the camera at constant x-y intervals, to create a regularly sampled measurement 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.

Refocusing an unstructured lightfield. (50 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 the refocusing process: 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, the sampling grid structure is not known. 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.

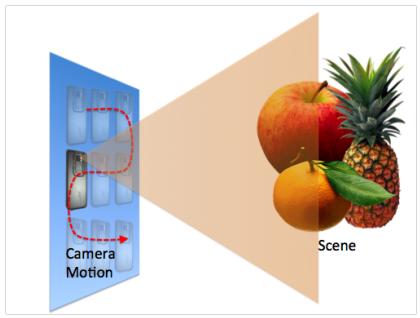

Figure 5: Capturing an unstructured lightfield.

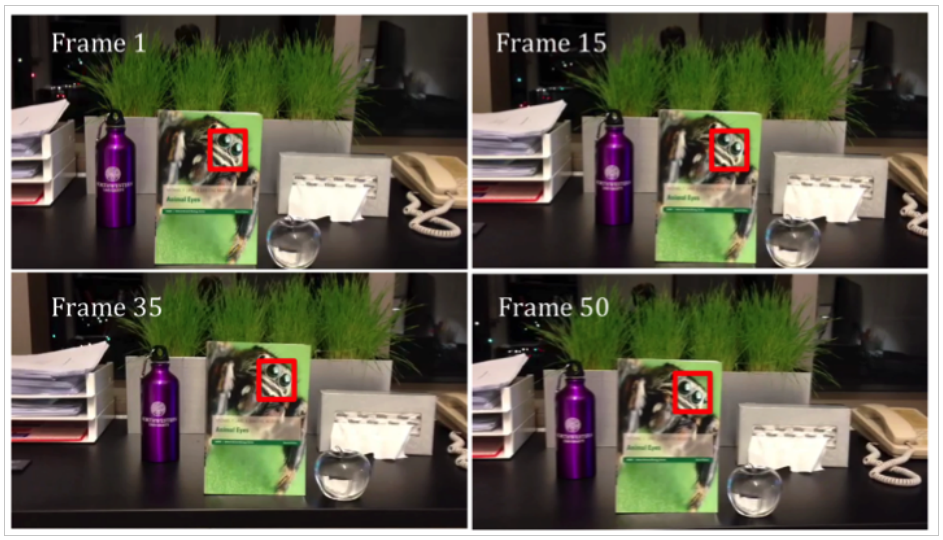

Figure 6: Frames from an unstructured lightfield.

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 frames of the video. Use this template to perform template matching on all other frames of your video, using normalized cross-correlation method: 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 is computed as:

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

where $\bar{I}_t[i,j]$ is a version of $I_t[i,j]$ filtered with a box filter of the same size as the template $g[i,j]$. The left part of Figure 7 shows a visualization of this. Note that template matching should be performed using a grayscale template and on grayscale video frames (you can use the luminance channel of the template and frames for this).

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

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

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

4. Bonus: Capture and process your own focal stack (60 points)

As we discussed in class, you do not need a lightfield camera to do depth from defocus or create all-in-focus images. Instead, you can do the same by using your regular camera to capture a focal 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 stack, then process it to create a 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 use the largest aperture available to your lens, as the shallow depth of field this will create will increase your depth resolution. We strongly recommend that you capture your focal stack by tethering your camera to your laptop, and using `gphoto2` or some other software to control it remotely.

Additionally, depending on the depth range captured in your focal stack, you will likely need to first perform an alignment step on your focal stack, to account for the change in magnification as the focus changes. 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 aperture and focusing distance settings reported by the camera to figure out the exact scaling you need to use.

Deliverables

As described on the course website, solutions are submitted through Canvas. Your solution should be an archive (e.g., a ZIP file) that includes the following:

- A PDF report explaining what you did for each problem, including answers to all questions asked throughout problems 1 and 2. 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 Matlab code, as well as a `README` file explaining how to use the code.
- The viewpoint collage you create for problem 1, refocused images that you create for problems 1 and 2, as well as the video you use for problem 2. 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-defocus 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.

Please organize your solution submission using the following file structure:

```
.zip
├── .pdf ............................................................ The PDF report.
├── src/ ............... Contains all Matlab M-files and the README file explaining how to use the code.
└── data/ ........................................... Contains all image, video, and other data files.
```

Hints and Information

- To implement the shifts described in various parts of the homework, you can use the Matlab function `interp2` that you are already familiar with from Homework 1.

- 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 = (1:lensletSize) - 1 - maxUV;
v = (1:lensletSize) - 1 - maxUV;
```

- The quality of your results in problem 2 will critically depend on how you moved the camera while capturing the video. In particular, you need to make sure that You avoid tilting and rotating the camera as much as possible. You should just shift the camera in a plane, as shown in Figure 5.

  When capturing the unstructued 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 more you cover the plane, the better your refocusing results in the next part of the problem 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 of 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).

- The template-matching-based refocusing in problem 2 is a very computationally intensive operation. Below are a few tips for making this faster.

  Make sure to implement normalized cross-correlation using convolution (e.g., using Matlab's `imfilter`). The Szeliski textbook explains how this can be done in its section about image alignemnt.

You can use all of 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 sequence 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 of results 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 3, the template was 60×60 pixels and the maximum shift of the template was about 40 pixels, therefore a search window of 200×200 pixels was used.

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 2 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] 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.