Image and Video Processing
Laboratory 4
Light Field Imaging

Dr. Martin Řeřábek, Irene Viola, Prof. Touradj Ebrahimi

Reports submission – !!! read carefully !!! Students will have to produce and submit a report two weeks after completion of each laboratory session. An electronic version of the report is submitted via Moodle's interface for uploading assignments. There will be a hard deadline for each submission. Late submissions will have to be justified and explained to responsible assistant by email. A structure of the report is explained below. In addition to the report, the source code produced should also be submitted in electronic form. The report and the source code should be submitted in **ONE ZIP** file using Moodle platform.

Structure of the reports and m-files

- the name of the final ZIP file will be: `lab_`*number-of-lab_your-surname*`.zip`
- the final ZIP file will contain
  - final m-file called `run_lab_`*number-of-lab_your-surname*`.m`
  - all created m-files and functions which are necessary to obtain the results
  - all source images and data needed for successful running of the final m-file
  - an electronic version of your report in **PDF** format
- the structure of the final m-file
  - the final m-file will contain all necessary commands and functions, by running this m-file one must get all results[1]
  - submitted m-files will be commented
  - each figure will be properly titled
  - the final m-file will be divided into cells[2] according to the example bellow
- the parts of program codes should not be included in the final report
- description of the results should be a part of the final report
- don't forget to answer all the questions in your report
- check whether your final m-file can be launched without problems - only then submit your report

[1]of course it is upon you whether you want to include everything in the final m-file, or create different functions which you will call in the final m-file

[2]those who do not know how to use the cells in Matlab to create the program code more transparent and easier to read, they should look at `http://www.mathworks.com/demos/matlab/developing-code-rapidly-with-cells-matlab-video-tutorial.html`

An example of final m-file

```
% Lab (number of lab) - your name and date

%% Exercise (number of exercise) - "Name of exercise"

%your own program code with your coments
a = imread('picture.tif');
%all figures will be properly titled
figure('name','Name Of The Figure')
imshow(a,[]);

%% Exercise (number of exercise) - "Name of exercise"

%your own program code with your coments
[output1, output2, .., outputN] = function_name (input1, input2,..,inputN);
```

0 Introduction

Background information on LF imaging and acquisition and rendering and visualization will be given during the lecture in the beginning of lab 4 session. Further details on LF imaging can be found for example in [1].

1 Participation on the subjective evaluation of LF content

Without this task you will not be able to successfully finish the lab 4. In this exercise, the task consists in performing a short subjective experiment in semi-controlled environment. This exercise will help you to understand how LF image can be rendered and what artifacts are related to LF image rendering. It will also help us to collect real raw scores for further studies. The collected data won't be analyzed within this lab session.

1.1 Tasks

1. Log in one of the computer in CO5.
2. Set up your display resolution to 1920x1200 pixels.
3. Start chrome browser.
4. Make sure the window of browser is maximized, i.e., the web browser is spanning the whole screen area. Pressing F11 will do it for you.
5. In the browser, go to `http://grebvm2.epfl.ch/Lab4/index.php` and follow the instructions to complete two short sessions of subjective experiments.

2 Image Data

For the following exercises, you will use three different light field image content, namely Bikes, Friend, and Fountain. Light field images can be downloaded from moodle in 4D LF format suitable for import to Matlab.

3 Refocusing of LF Images

One of the main features offered by light field acquisition is the possibility of rendering the scene with different focal points. For example, it is possible to render the scene in such a way that only one plane in the scene is in focus, while the other planes are blurred. This is referred to as "synthetic aperture photography". One simple way to digitally refocus the scene is by means of a **shift and sum** filter which is applied to 4D light field image. The idea behind it is to shift all perspective views to a common depth. After all the views have been shifted, in order to obtain a refocused image, all the individual perspective views have to be added together. The plane on which the image will be refocused depends on how much each view is shifted. For any view at position (i, j) the shift p_i and p_j in pixels is computed as

$$\begin{aligned} p_i &= X[-\frac{1}{2} + \frac{i-1}{N-1}] \\ p_j &= X[-\frac{1}{2} + \frac{j-1}{N-1}], \end{aligned} \tag{1}$$

in which N is the total number of views (in our case, $N = 15$) and X is the slope parameter.

The views $V_{(i,j)}$ need to be shifted accordingly as follows

$$V_{(i,j)}(k, l) = V_{(i,j)}(k + p_i, l + p_j). \tag{2}$$

The value of p_i and p_j is rarely an integer, thus an interpolation needs to be used to compute image values at positions corresponding to desired shifts. After shifting, the individual views are summed to obtain a refocused image as follows

$$O(k, l) = \sum_{i=1}^{N} \sum_{j=1}^{N} V_{(i,j)}(k, l). \tag{3}$$

After summing, the values of $O_{(k,l)}$ need to be normalized in order to be visualized.

3.1 Tasks

1. Write a Matlab function that performs the refocusing. The function will get as a parameter the 4D LF image, which will be given to you, as well as the slope parameter X. The output will be the refocused image. Use slope values in range $\langle -10, 10\rangle$.
2. **BONUS:** Implement another method to refocus the LF image.
3. Evaluate your results.

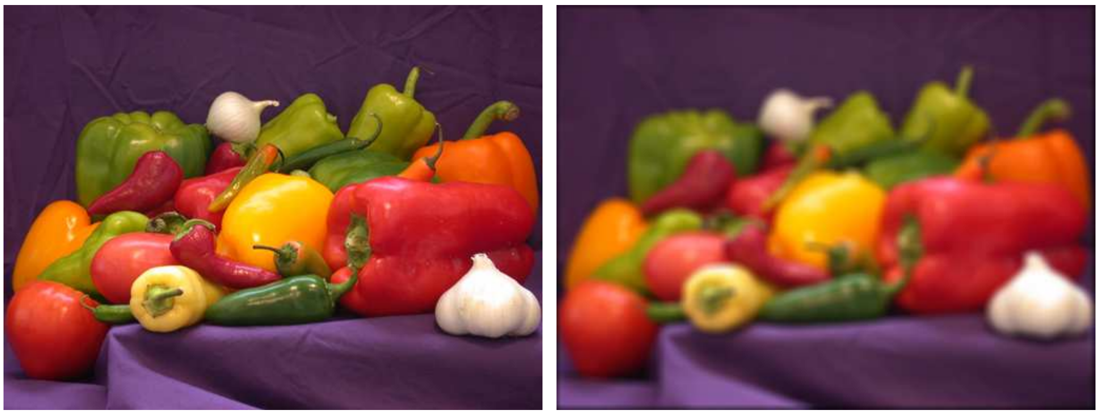
NOTE: useful Matlab commands: `interpn`

4 Depth of Field of LF Images

Depth of Field vs Blur

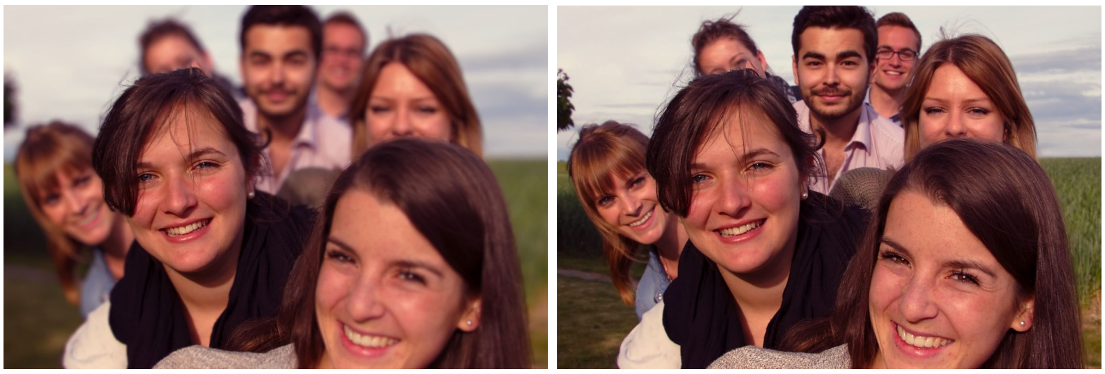
An image appears blurred when its high spatial frequency values in the spectrum are attenuated. Different types of blurs exist. For example, motion blur due to the relative motion between the camera and the scene, and out of focus blur due to a defocused camera and lens aberrations. Blur can also be introduced when processing the image data, such as performing compression or filtering. A visual example of blurred image is shown in Figure 1.

Depth of Field, on the other hand, is not a convolution of the image. It originates in nature of light as a wave and it is related to a circle of confusion variation with depth. Theoretically, a camera can effectively



(a) (b)

Figure 1: Example of (a) original and (b) blurred image.



(a) (b)

Figure 2: Example of (a) f/1 and (b) f/8 aperture.

focus only at one focal plane at a time. However, in practice, taking into account the limitations of human visual system, one can perceive a range in image plane which appears sharp. The perceived sharpness is determined by size of the aperture, commonly referred to as f-number. Example of images taken with two different apertures is shown in Figure 2.

Aperture Size

Depth of Field (DoF) is the distance between the nearest and the farthest objects giving a focused image. In photography, one can choose to have a narrow or large depth of field by changing the aperture of the camera. A very small aperture will give a very large DoF, while a very large aperture will results in a very narrow DoF.

The DoF of a light field scene can be changed arbitrarily, since it depends on the number of views that are used in the shift and sum filter. For example, using only the 9×9 central views will result in a larger DoF than choosing the central 13×13.

4.1 Tasks

1. Modify the Matlab function created in the previous exercise. The new function will have a new parameter A, which defines the aperture of the scene. Apply for values of A in the range of [1, 15]. Value $A = 3$ means that central 3×3 images will be used.

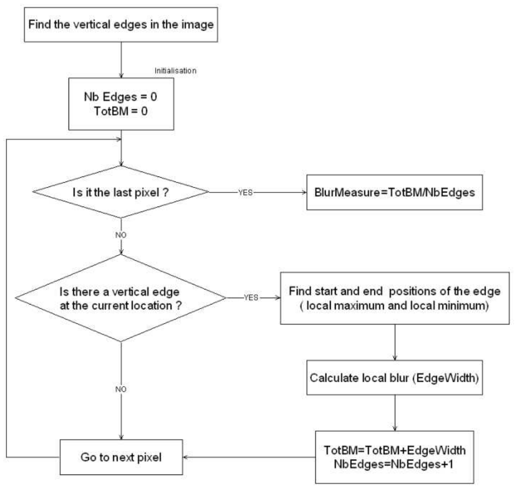
2. Evaluate your results.

5 Depth of Field Measurement

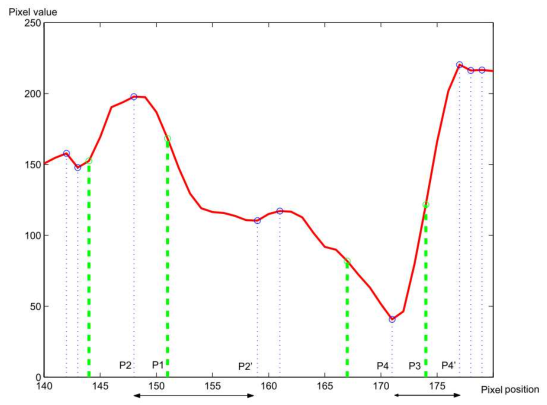
No-Reference blur metric

In this exercise you are asked to implement the blur metric proposed by Marziliano et al. [1] and to evaluate the influence of aperture size on the amount of "blur" in an image. No-Reference blur measurement technique assumes no knowledge of the original image and does not make any assumptions on the type of content or the blurring process. The blur measurement is defined in the spatial domain. Starting from the observation that blur is perceptually apparent along edges or in textured areas, the technique is based on the smoothing effect of blur on edges, and consequently attempts to measure the spread of the edges. The blur is usually measured along vertical edges only, however you are required to measure the blur along both axis. For color images, blur is measured on the luminance component Y. The algorithm for vertical direction is summarized in Figure 3(a), and consists of the following steps:

1. First an edge detector (e.g. Sobel filter) is applied in order to find the edges in the image.
2. Then, each row of the image is scanned: for each pixel corresponding to an edge location, the start and end positions of the edge are defined as the local extrema locations (i.e. pixels corresponding to local minimum and local maximum) closest to the edge. The edge width for each pixel corresponding to edge location is then given by the difference between the end and start positions, and is identified as the local blur measure for this edge location. An example of a row in a image is illustrated in Figure 3(b). For the edge location P1, the local maximum P2 defines the start position, while the local minimum P2' corresponds to the end position. The edge width is P2'-P2 or 11 pixels for this example. Similarly, for the edge P3, the local minimum P4 is the start position, the local maximum P4' is the end position, and P4' - P4 is the edge width.
3. Finally, the global blur measure for the whole image is obtained by averaging the local blur values over all edge locations.


Find the vertical edges in the image
Initialisation
Nb Edges = 0
TotBM = 0
Is it the last pixel ?
YES
BlurMeasure=TotBM/NbEdges
NO
Is there a vertical edge at the current location ?
YES
Find start and end positions of the edge (local maximum and local minimum)
Calculate local blur (EdgeWidth)
NO
Go to next pixel
TotBM=TotBM+EdgeWidth
NbEdges=NbEdges+1

(a)



(b)

Figure 3: (a) Flow chart for No-Reference blur metric. TotBM denotes the total blur measurement, NbEdges denotes the number of edges. (b) Pixel values along one row of the luminance component of a test image. The detected edges are indicated by the dashed green lines, and local minima and maxima around the edge by dotted blue lines. The edge width for edge location P1 is P2' - P2.

5.1 Tasks

1. Write a function `[blur_index_ver,blur_index_hor] = NR_blur(Y)` which takes as input the luminance component of an image (`Y`) and computes as output the blur index (`blur_index`)for vertical and horizontal direction, according to the method described above and detailed in Figure 3.
2. Compute the blur index as an average between vertical and horizontal blur index values for following cases
   - content Friends - slope $X = -4$, aperture $A = \{3, 5, 7, 9, 11, 13\}$;
   - content Bikes - slope $X = 5$, aperture $A = \{3, 5, 7, 9, 11, 13\}$;
   - content Fountain - slope $X = 5$, aperture $A = \{3, 5, 7, 9, 11, 13\}$;
3. Plot the dependencies between aperture size and blur index.
4. Evaluate your results.

NOTE: useful Matlab commands: `edge, rgb2ycbcr`

References

[1] R. Ng, M. Levoy, M. Brédif, G. Duval, M. Horowitz, and P. Hanrahan, "Light field photography with a hand-held plenoptic camera," *Computer Science Technical Report CSTR*, vol. 2, no. 11, pp. 1–11, 2005.

[2] P. Marziliano, F. Dufaux, S. Winkler, and T. Ebrahimi, "A no-reference perceptual blur metric," in *Image Processing. 2002. Proceedings. 2002 International Conference on*, vol. 3. IEEE, 2002, pp. III–57.