UNIVERSITÄT TÜBINGEN
LEHRSTUHL FÜR COMPUTERGRAFIK
PROF. DR.-ING. HENDRIK P.A. LENSCH
JIEEN CHEN

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

24.05.2018

COMPUTATIONAL PHOTOGRAPHY
ASSIGNMENT 4

Submission deadline for the exercises: Thursday, 31. May 2017 before 11:59.

Instructions: Upload the source code to your solution **(no images please, just the plain code)** in the ILIAS system at:
ILIAS Computational Photography SoSe2018
This assignment sheet has a total of 90 points.

4.1 Depth of Field (90 Points: 5, 30, 5, 5, 30, 5, 10)

Camera optics and lens effects are usually neglected when implementing a simple raytracer, so all it produces are images that are focused everywhere and free from lens abberations and it is not possible for the user to achieve a photographic look. However, it is very easy to extract depth information from a raytracer.
In this exercise, you are going to create a synthetic Depth-of-Field (DoF) effect into a rendered picture of the famous Sponza Atrium in the Šibenik cathedral. To do so, you are given two images: an all-in-focus image of the Atrium and the depth map for each pixel. In order to create a proper looking picture, you need to add some DoF blur to it.
The focal length of the lens, radius of the lens aperture and the sensor size are provided. The desired focusing distance is 1.

(a) All-in-focus image (b) Depth image (c) Correctly ray traced image

Figure 1: Given images for creating image-based DoF, and the correct ray traced image.

Your task is to implement the underlying functionality:

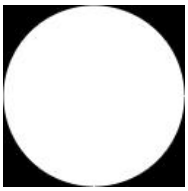
a) Compute the distance from the lens plane to the image plane (distance between the lens and the sensor).

b) For each pixel precompute the radius of the blur kernel in the image plane (as a floating point value).

c) Pad the image `img` by replicating the border around the image such that blurring with the largest kernel can be done even on the border of the original image. (**Hint:** `padarray` function might be useful.)

d) In case there is no blurring for the current pixel fill in the values for the `blurImg` and the `accumBuff` appropriately.

e) In case there is blurring perform the splatting of the pixel value to its neighbors and update the `accumBuff`. The splatting is to be performed using the given radial blur kernel `kernel`. (**Hint:** `blur_kernel` is a quick implementation of `fspecial('disk',size)`.)



Radial blur kernel.

f) Finally, normalize the image using the `accumBuff`.

g) Compare the result you obtained with the provided image which has been correctly ray traced using a thin lens model - why are there differences?

Hints:

- The given depth map contains the ray traced depth from the point of view to the object in the scene the ray intersects. Even though this is not the depth of the plane parallel to the image plane which contains that object point (which would be needed for correct calculation of the DoF), we will assume that it is.
- Pixel splatting is basically accumulation of the value of the current pixel to its neighbors. If a splatting kernel is given, then the splat of the current pixel for the given kernel is a component-wise accumulation of the kernel to the neighborhood, scaled by the pixel value.
- Don't be surprised if the code takes a while to run (you might want to test it on downscaled images).

General Tips:

- Instead of working on 3D-matrices all the time you can extract the color layers to separate variables and work with 3 2D-matrices, or you can index every channel separately. This saves you some stress with indexing errors.
- For the star exercise you can submit your explanation in one of the following ways:
  - Add it as a comment in the code (at the beginning of the 'dof.m' file).
  - Write a PDF/txt file containing the explanation.

Links:

- Depth of field is nicely explained at
  http://www.cambridgeincolour.com/tutorials/depth-of-field.htm