Final Project: Depth of Field

Miguel Neves dos Reis Prazeres 6079067
Technische Universiteit Delft , Netherlands

1. Introduction

In this project a depth of field effect was developed based on diffused scribbles. My approach consists of allowing the user to scribble depth annotations in a python UI app (using tkinter). Then those scribbles in a RGBA image are loaded into the c++ program (inspired in the code of the assignments). In c++ this algorithm is implemented to calculate the depth [1]:

Poisson Diffusion

Given the image $I := \{I_{i,j} \mid i \in 1..w, j \in 1..h\}$, where $I_{i,j}$ are brightness or color values at pixel (i, j), we aim at creating a depth map $D := \{D_{i,j} \mid i \in 1..w, j \in 1..h\}$, given a set of scribbles with associated values $\{S_{i,j} \mid i \in 1..w, j \in 1..h\}$ on these scribbles. The depth map D is then implicitly defined:

$$\Delta D = 0$$
$$\text{subject to: } D_{i,j} = S_{i,j}, \forall (i, j) \in I$$

Fig 1 – Poisson Diffusion of scribbles

straints. For a pixel k and its 4-pixel neighborhood $N(k)$, we obtain:

$$\sum_{l \in N(k)} \omega_{kl}(D_k - D_l) = 0, \qquad (2)$$

where ω_{kl} is the first order difference for the two neighboring pixels $\omega_{kl} = \exp(-\beta|I_k - I_l|)$. At the border of an

Fig 2 – Anisotropic Diffusion using weights

First, I calculate the respective weights in the function *calculateWeights* in the file *your_code_here.h.* Then the Poisson equation in Fig 2 is solved using a similar method used in assignment 1, but with the weights.

Finally, to simulate the depth-of-field, I calculate the Circle of Confusion for each pixel, and apply a jointBilateralFilter (based on assignment 3).

The Circle of Confusion is calculated using this formula [2]:

$$c = d \times \left| \frac{v_p - v_o}{v_p} \right|.$$

Formula 1 - Circle of Confusion

Where v_p is the value of the current pixel, v_0 is the focus depth and d is the aperture size. The focus depth takes integer values from 0 to 255 (range of RGB image), and aperture size takes integer values from 1 to 10. Therefore, the depth value with the depth focus is the layer that is focused, and the bigger the aperture size the bigger the blur in the image.

I implemented all the mandatory features and the pretrained RGB->Depth CNN using MiDaS [3].

Here are some results:

Image 1 – arch source

Image 2 – lotus source

Image 3 – arch scribbles

Image 4 – arch diffused scribbles

Image 5 – arch depth-of-field with focus depth of 200 and aperture size of 2

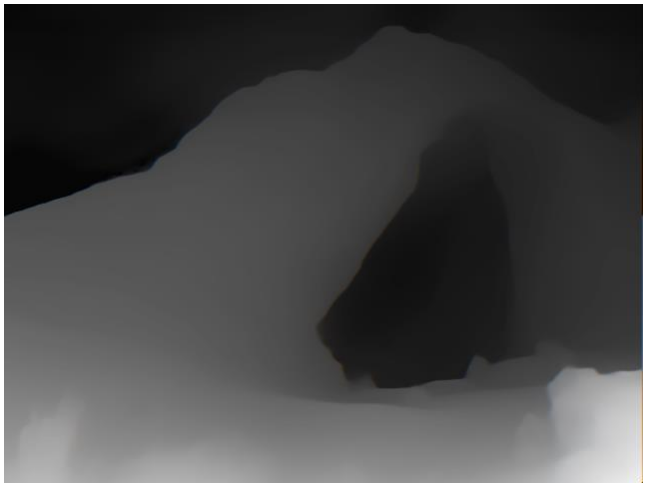

Image 6 – arch depth estimated by MiDaS (grey scale)

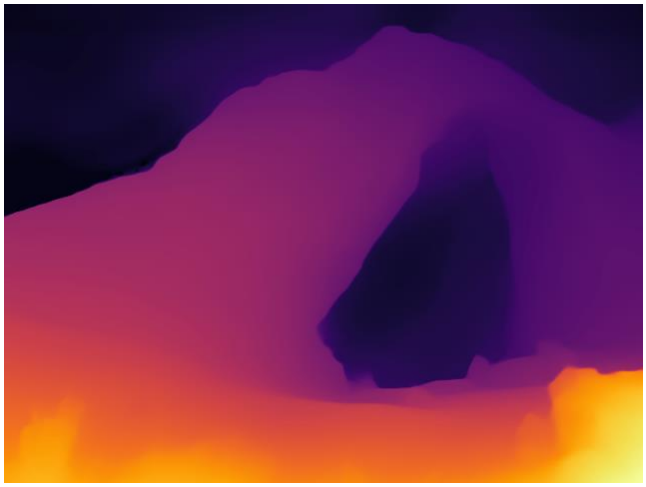

Image 7 – arch depth estimated by MiDaS (original)

Image 8 – arch depth-of-field with focus depth of 200 and aperture size of 6 (using greyscale depth of MiDaS)

Image 9 – lotus scribbles

Image 10 – lotus scribbles diffused

Image 11 – lotus depth-of-field with focus depth 200 and aperture size 2

Image 12 – lotus depth estimated by MiDaS (greyscale)

Image 13 – lotus depth-of-field with focus depth 10 and aperture size 8 (using greyscale depth of MiDaS)

References

[1] Liao, Jingtang, Shuheng Shen, and Elmar Eisemann: "Depth annotations: Designing depth of a single image for depth-based effects." Computers & Graphics (2018). https://graphics.tudelft.nl/Publications-new/2017/LSE17a/depthannotations-authorsversion.pdf

[2]https://developer.nvidia.com/gpugems/gpugems3/part-iv-image-effects/chapter-28-practical-post-process-depth-field

[3] https://github.com/isl-org/MiDaS