Computational Bokeh Effect Implementation Using Depth Estimation Image

CS445 Final Project. Mingzhong Sun, Yue Chen

Motivation and impact:

Previously we have learned some very basic ideas of algorithms that can predict the depth information in a single image by detecting edges and calculate depth accordingly. We then started to explore the applications of depth estimation of a single image. Initially, we thought predicting depth information from screenshots of Minecraft [5] and compute background blurring according to different depth levels could be interesting. We subsequently recognized that this approach could be applied to any image, thereby significantly broadening its potential impact.

This idea brought us to an area of computational bokeh effect of images. There are many real-life examples. Smartphone companies have been trying to mimic DSLR camera shots with these algorithms in recent years. Adobe Lightroom is now testing their new “lens blur” function [6]. We notice that this is indeed what people need to make their photo better – how to “fake” these photos in a way to make them indistinguishable from DSLR photos is what people are currently aiming for.

While progressing with our project, we discovered the existence of an annual Bokeh Challenge [1] hosted by NTIRE (New Trends in Image Restoration and Enhancement) and CVPR (Conference on Computer Vision and Pattern Recognition). It seems that Inspired by the 2020 paper 'Rendering Natural Camera Bokeh Effect with Deep Learning,' [2] the deep learning approach for Bokeh rendering is now regarded as more advanced than traditional computational photography techniques, thus widely applied in both such competition and industry applications. Thus, upon completing our implementation with the traditional approach, we plan to conduct a comparative analysis of the output images against those produced by deep learning methods, and image processing softwares (Adobe Lightroom), with photos filmed by professional camera serving as a benchmark. This will help us determine if traditional computational photography approaches for Bokeh rendering are a dead end.

Approach:

The first step of our approach is to obtain the depth estimation information of the image. We’re using MiDaS v3.1, which is the latest Monocular Visual-Inertial Depth Estimation model released by Intel Labs in December 2022 [2]. We anticipate that this state-of-the-art model will deliver highly accurate depth estimations, thereby laying a solid foundation for our subsequent creation of the Bokeh effect.

After we have the depth image, we have tried two ways to compute the bokeh/blur. Bellow are some abstract concepts of our implementations:

Given a depth image, we want to classify it to a constant number of levels of depth. This could be a parameter. Given a focus point, we assign areas closer to the focus point to be lower levels, and let deeper areas have higher levels.

Part 1: The naive approach

- Create a few copies of the original color image and blur them using Gaussian Blur using different kernel sizes.
- For the output image, for each pixel, lookup and copy the corresponding blurred pixel from blurred images.

We then discover that a serious problem is making the blurring unrealistic. Outside the border of the focused object, if the background is close to the focused area, foreground color is used for calculating Gaussian blurring of the background image. This results in a feathering of the focused object on its border, which is not how a DSLR photo looks like. Here is an image of the affected area:

With that in mind, we came up with the second approach.

Part 2: Using inpainting.

- For each depth level, create a mask that removes lower-level areas.
- Use hole filling/inpainting methods to fill the hole.
- Blur the filled images and create the output as what we have done in the naive approach.

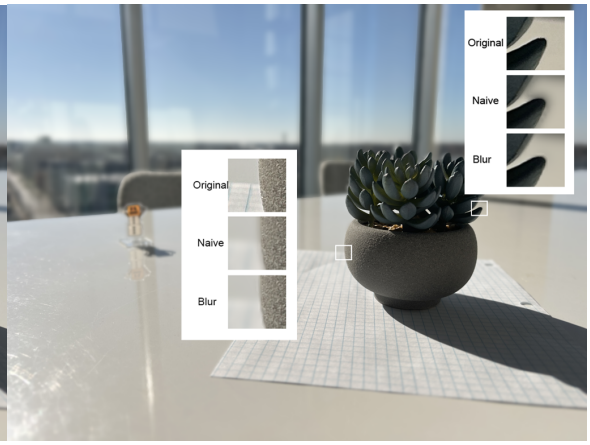
We came up with this idea that filling the hole will make the Gaussian Blurring more consistent with the color in its own level. Since we don't require too much information from textures, we can use a simpler algorithm that only paints the color and intensity from the area outside the hole.

Results:

Image Input

Naive Blurring

Better Blurring

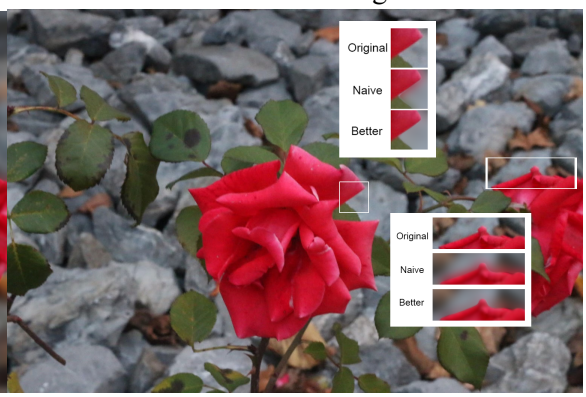


Comparison

Image Input

Naive Blurring

Better Blurring



Comparison

Deep Learning

Adobe Lightroom

Canon 7D

When focusing on the “villager”:

Input Image

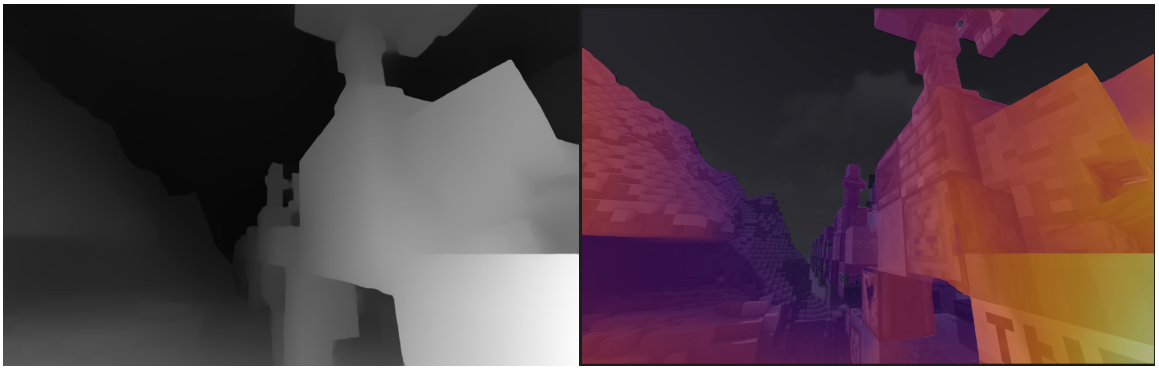
Adobe Lightroom

Better Blurring

Analysis:
For the comparison between our own algorithms, we can clearly see that there is an enhancement for the "feathering effect". However, there are indeed some limitations of the "better approach". We noticed that, when some of the background has huge differences in relative depth, the boundary of the inpainted region is obvious. However, if the relative depth is huge while both levels are deep, the boundary should not be obvious. We should leave them blurred, or to make them more realistic, they should be blurred using some other methods to create different bokeh effects. Due to time limitations, we could not research these effects.

In the flower image, we see that deep learning is better than Adobe's and our results. This could happen as we (also Lr.) are relying on depth estimation for blurring the image. This suggests that as long as there is limited information in the image to correctly estimate the relative depth, we will not be able to produce a correct image that is close to the DSLR image.

Comparing results on the Minecraft screenshot, we notice that Adobe is not working well on images that have the following requirements: there exists foreground, midground, and background. When focusing on the objects in the midground, sometimes Lightroom cannot correctly blur the background – even though Lightroom is indeed using a depth image. In the image above, we do see that the mountain in the back is not blurred, and the sky is not changed. However, this is just our observation for this image. In general, Lightroom is doing a great job at computing a good blurring, and they are very good at dealing with edges of objects. We provide depth images for the Minecraft screenshot.



MiDaS Depth Image Lightroom "Visualize Depth"

The problem showed up in the Lightroom image could be caused by incorrect depth image estimation. But still, they are not correctly dealing with the sky (the area that has infinite distance).

Implementation details:

We choose to use python as our programming language. Here are some libraries we used:

- CV2(OpenCV)
- matplotlib.pyplot
- numpy

Open source:

(you can download model and source code from our google drive below instead of github)
MiDaS v3.1
Midas3.1.ipynb
https://drive.google.com/drive/folders/1KzDET2ipPX1CtqHxakwj9Gzyof1HvAhk?usp=sharing

Input image:

Most taken with our own iPhone. Some images from paper "Rendering Natural Camera Bokeh Effect with Deep Learning"
Implementation details of bokeh rendering can be seen from: https://github.com/MingzhongSun/-cs-445-Monocular-Depth-Estimation-for-Minecraft

Challenge / innovation:

We expect to receive full credit for innovation part with following challenges and innovations:

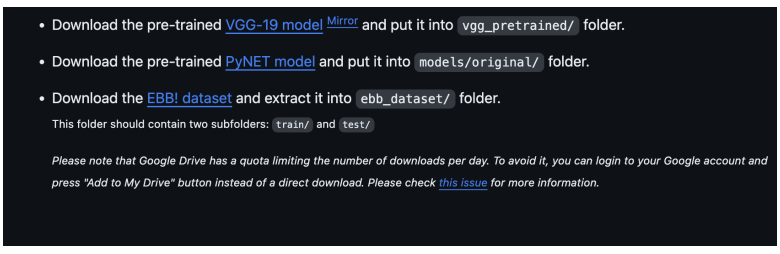
Challenges:

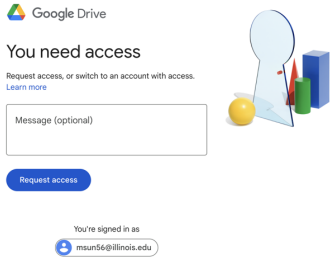
1. The blurring algorithm we developed is entirely original, having evolved progressively based on the outcomes observed during the implementation phase. It was not derived from or influenced by existing literature. While implementing, we do discover that there are some limitations of our algorithm. This was discussed in the analysis in our results section.

2. Implementing open-source models, such as MiDaS v3.1 and Bokeh Rendering models, proved more challenging than anticipated.

For MiDaS, the widely used and mature inference techniques in Google Colab don't align with the latest v3.1 model. Fortunately, we were able to successfully implement MiDaS v3.1 after making several necessary revisions in source code.

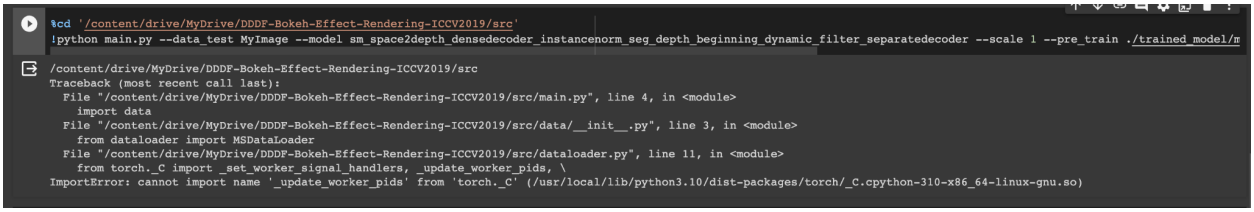
As for the Bokeh Rendering models, we attempted to implement the PyNET-Bokeh model from 'Rendering Natural Camera Bokeh Effect with Deep Learning.' Unfortunately, we faced accessibility issues with the pretrained PyNET model, hindering our implementation efforts.





We then shifted our focus to another deep learning model for Bokeh rendering from the 2019 ICCV paper 'Depth-Guided Dense Dynamic Filtering Network for Bokeh Effect Rendering.' [4] Given the several

years' gap since its publication, we encountered numerous library compatibility issues during debugging. After debugging hopefully for several hours, a critical problem emerged regarding the PyTorch version: the paper's implementation was based on version 1.8.0, whereas the oldest version supported by Colab is 1.11.0. This version discrepancy led to significant challenges, as a range of libraries required for the model could not be imported and were hard to transform to new libraries.



Due to the extensive time spent on debugging these models, which far exceeded our initial estimate of 15-20 hours, we ultimately had to abandon the task of inferring these deep learning methods from scratch. Instead, we resorted to directly comparing the results presented in their papers with our outputs.

Innovations:

1. From our knowledge we haven't found any implementations in the field that combine the latest state of art Depth Estimation models (MiDaS v3.1) with the traditional Computational Photography blurring methods. Hence, this novel approach and its resulting output have the potential to offer fresh contributions to the field. And based on our result, the MiDaS v3.1 does show more accurate depth estimation results compared to Adobe lightroom depth estimation tools in some cases (Minecraft).

2. Based on comparison between deep learning approach and traditional approach, we found that the deep learning approach lacks the way to customize the blurring parameters and blurring degree. The way of how they compute blurring depends on the fixed training sets. There are more demands in the world now that require adjustments for computing the bokeh interactively and more precisely. This is not saying that full automation is bad, but it is more common for people to adjust the parameters as they wish.

Future work:

As we discussed in the analysis of our result, the limitation of our better approach should be fixed in future. Research on other blurring methods is required. We also propose a way to remove this effect on the background. Suppose we only want to limit the feathering around lower level areas, we can try applying edge detection on objects in the lower levels. Then apply the weighted average of the results from two approaches.
Also, notice that our calculation takes longer than other existing commercial algorithms. So optimizations are also required if user experience is important for you.

References:

[1] https://codalab.lisn.upsaclay.fr/competitions/10229

[2] Andrey Ignatov, et *Rendering Natural Camera Bokeh Effect with Deep Learning* https://ar5iv.labs.arxiv.org/html/2006.05698

[3] *Intel Labs Releases Models for Computer Vision Depth Estimation: VI-Depth 1.0 and MiDaS 3.1* https://community.intel.com/t5/Blogs/Tech-Innovation/Artificial-Intelligence-AI/Intel-Labs-Releases-Models-for-Computer-Vision-Depth-Estimation/post/1468771

[4] Kuldeep Purohit, et *Depth-Guided Dense Dynamic Filtering Network for Bokeh Effect Rendering* https://ieeexplore.ieee.org/abstract/document/9022538

[5] Minecraft, https://www.minecraft.net/

[6] Lens Blur in Lightroom, https://helpx.adobe.com/lightroom-cc/using/lens-blur.html