Artificial Bokeh from a Monocular Viewpoint

Alex Bainbriddge
Carnegie Mellon University
abainbri at andrew.cmu.edu

Abstract

This work focused on how bokeh can be created and applied to images artificially. It utilized advancements made in machine learning, depth estimation techniques, and work in computational photography. The basis for this work is Barron et. Al's Fast Bilateral-Space Stereo for Synthetic Defocus. [1] By utilizing the depth map approximation, we are able to estimate a bokeh effect upon the desired image while taking into account an applied alpha for error correction.

1. Introduction

Many of todays amateur photographers need to create artistic photos on a budget. One of the convenient ways this can be achieved is through the use of post processing toolkits. These toolkits provide a variety of features such as blurring, bokeh, patching, and many more. The creation of these toolkits fall under computational photography, and are an ever-growing research field. As well as generating new toolkits, there are also active efforts to improve existing ones and to use new approaches in order to get better results or make the tools easier/simpler to use. This lowers the barrier for entry and allows more people to get involved in the field of photography.

2. Problem Statement

One of the most commonly desired artistic effects is bokeh. This effect is a blurring of the background in order to give a focused and clean look to the foreground. There have been various other attempts to perform this, however most rely on either a ground truth depth map, or with a stereo image pair. However, these complex imaging techniques can be hard to capture and make these methods for artificial bokeh difficult to utilize. A simpler method for artificial bokeh is needed in order to let more people have access to artificial bokeh.

3. Background Information

Currently, there are many different approaches to apply artificial bokeh in post processing. Two of the most popular methods utilize either a measured depth ground truth map, or stereo pairs to infer a depth ground truth map. The second method has gained wide spread adoption recently due to Google's deployment in its new camera technology. [1]

One common approach in todays world, is to use machine learning to tackle difficult problems where there is a wealth of data and information. Luckily, computational photography problems often have a wealth of information to mine for patterns. As such, Saxena et al. have developed a depth estimator for monocular images. Their work was crucial to my work as it provided a solid foundation for generating approximate depth maps. They utilized indoor and outdoor scenes and generated various models on these scenes in order to cover a wide variety of applications and use cases.

4. Approach

4.1. Depth Estimation

The first goal was to connect the make3d model to matlab. This would provide the basis for the depth map creation needed for the defocusing algorithm. After looking through all of the documentation, I was able to create the necessary matlab calls to generate the depth map. This is shown in mydepthmap.m on my github - https://github.com/Dreadwall/Artificial-Bokeh-Matlab-. Looking over the depth maps, and what they were supposed to be, they seemed to be relatively accurate and could be used for estimations.

4.2. Defocus Start Point

The next issue to solve, was how to determine where best to start off the defocusing. I decided to create a codebook of the pictures I planned to use, generated their depth maps, and mapped their maps to particular kernel sizes for the blurring filter. This codebook was useful as it grounded the problem of finding a mapping from depths to window-sizes to be purely a mathematical one rather than a subjective one. This process also allows for future work to perhaps create a better fitting start point, by data mining a large codebook of depth_map to window-size mappings. This would automate the process and greatly reduce the need for the error-correction alpha.

However, not being able to use such an approach, I manually attempted to find various mathematical expressions and equations that would fit well. Of those I found, I saved some of the better ones in defocus.m as comments for later viewing or tuning. While the cubic was one of the more accurate models for some pictures, it failed horribly on others. On the average case, the model using the median depth as the starting point faired the best. When combined with a user entered error-correction alpha, the median depth was able to provide a good and well distributed mapping of depths to window-sizes. This worked generally well because most photos seemed to have the subject stand normally close-by or towards the middle of their scene.

4.3. Defocusing

With the depth map finalized, and the starting point for the defocusing set. The algorithm begins to blur more for depths larger than the starting point and increasing based on its distance from the corrected median depth. After trying numerous bluring algorithms, one that respected edges and accounted for distance between pixels was needed so bilateral filter was used. Thus the three parts for the actual defocusing of pixel(i,j) was the window-size which was set by the depth at pixel(i,j), the neighbors of pixel(i,j) to detect edges, and the luminosities of all pixels in the window-size around pixel(i,j) for blurring. I also decided to set a max window-size to 15 pixels, as the image began to loose too much resolution beyond that. By nature of the window sizes and the depth maps, there is generally a smooth gradient in the scene - this allows us to avoid problems of harsh blur gradients in most cases.

To help speedup computation, we can pre-compute the gaussian weights for the various window-sizes and store them ahead of time. We also downsize the image to only around 500x600 in order to speedup the depth map estimation, which is one of the more costly operations of the algorithm. Also, we can utilize the computer's caches to speed up the depth estimation on later photographs. The first time we use the depth estimator, we have a run time of around 5 seconds, however on subsequent uses we only run for around half a second, showing a drastic improvement from keeping the depth model in memory.

5. Results

5.1. My Results

The results from the defocus seemed to work really well with the sample scenes that I tried. The sample results can be found in the appendix in figures 1 through 3. The first is the raw photo, the second is the depth map, and the third is the defocused photo. As you can see, there are no harsh changes in blurring, and all follow a nice blur gradient as the scene gets further away from the lens. The photos provided give off an artistic blur that draws significant focus to the foreground while not losing significant parts of the background.

Due to the ease of using the defocus code, which is just plug and play in matlab - it allows amateur photographers to be able to employe it with limited programming knowledge needed. It could be further interfaced with a file selector and an alpha value slider in order to help usability even more. It is also accurate enough to provide meaningful results, while it is still far off from the photos achieved from the stereo cameras - it is applicable to even older cameras and photos. Allowing new takes to be made on them, rather than trying to re-capture their shots.

In my results I focused on bokeh used in landscape scenes, this is because while traditional use of bokeh is for foreground / background pairs I wanted to expand this to include a new field of photographs. With the bokeh on landscape scenes, we can recreate the effect of distance blur we would experience if we were at the location the photo was taken. This provides photographers the ability to re-imagine their scenes as if they were back to the original place. One interesting find when exploring these landscape scenes, was that my algorithm also works to project artificial bokeh onto paintings. This meaning that artists can take on a new look of their painting by focusing it at different depths and exploring the new artistic effect the bokeh has on their work.

I played with various window size growths, thus making the blurring more or less aggressive. However, I started to get very hard edges between blurring depths. I didn't think this artifact would be acceptable, and thus moved to keep the growth only by 1.

5.2. Reproducing

I have included the sample files, code, and model that I used for my work on my github. It can be accessed at: https://github.com/Dreadwall/Artificial-Bokeh-Matlab-. By downloading this and setting it up in matlab, you should be able to run defocus.m and specifying the path to various images in the "Downloads" folder. The park, forest, and CMU images, depth maps, and defocused images shown in this paper, can be found on the github. Each one took approximately 5-20 seconds to run varying based on size and other factors. I suggest running it on CMU first to solve the cold start problem as stated above.

For example new-image = defocus$'../Downloads/CMU.jpg', 0$ should run defocus on the CMU.jpg file with an alpha correction of 0. I have included the various alpha values that I used as comments in defocus.m. In my experience these ranged from 0 to 10, but can be anything desired. For clarity, I have also incorporated the various depth maps that were produced as an intermediary step in the "Depth_Maps" folder, and the final results in the "Results" folder. To get an intermediary depth_map, run mydepthmap.m pointing towards an image in the downloads folder. An example call of this would be depth_map = mydepthmap$'../Downloads/CMU.jpg'$. This depth_map would need to be displayed using imagesc however.

6. Limitations

There are some issues with the defocusing approach when there are rapid changes to the depth of a scene. This can cause harsh blur edges in the result picture and make it look uneven. This could be solved with a better depth map estimator, or perhaps using a filter that considers how nearby pixels have been blurred. Another limitation for this project, was that I did not have access to a new gpu and could only rely upon my laptops cpu. This required that images be sized down for depth map estimation in order to achieve practical runtimes, however by doing this depth maps became less accurate as information was lost in the image resizing. There is also the limitation that only an outdoor scene set could be used with the provided dataset, and that with more information a more comprehensive depth model could be made to accept a whole variety of scene inputs.

7. Future Works

Future works in this space could involve placing the corrected median around a normal distribution, where the foreground in front of the median and the background are both blurred as they get further from the median. This may produce more interesting results as a particular depth level is now isolated completely from the rest of the scene.

Another interesting future work would be to incorporate multiple points of focus in the image - and two blur based on the distance from both points. This would allow pictures to have bokeh effects perhaps around two subjects or two parts of a scene, with an artistic effect around them both.

8. Appendix

Figure 1. Park Example

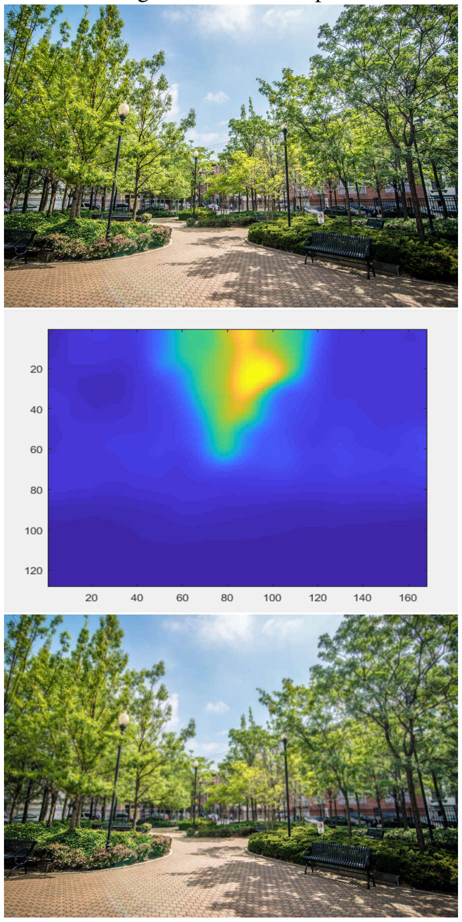


Figure 2. Forest Example

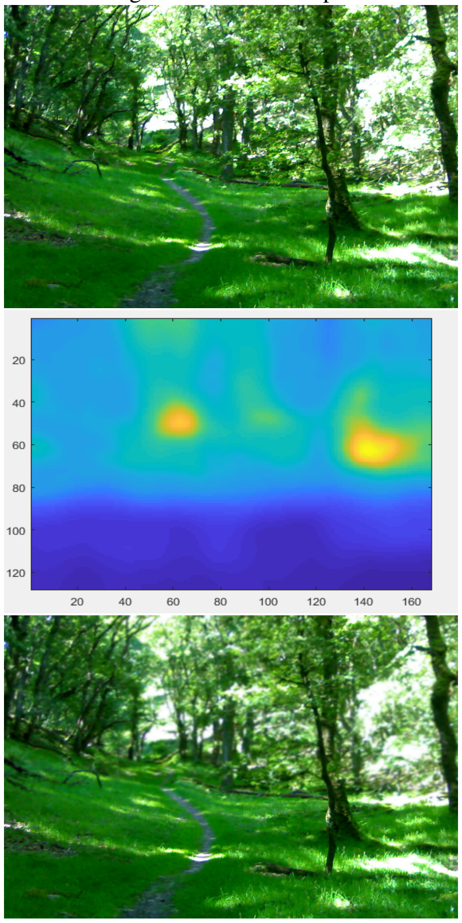


Figure 3. Forest Painting Example

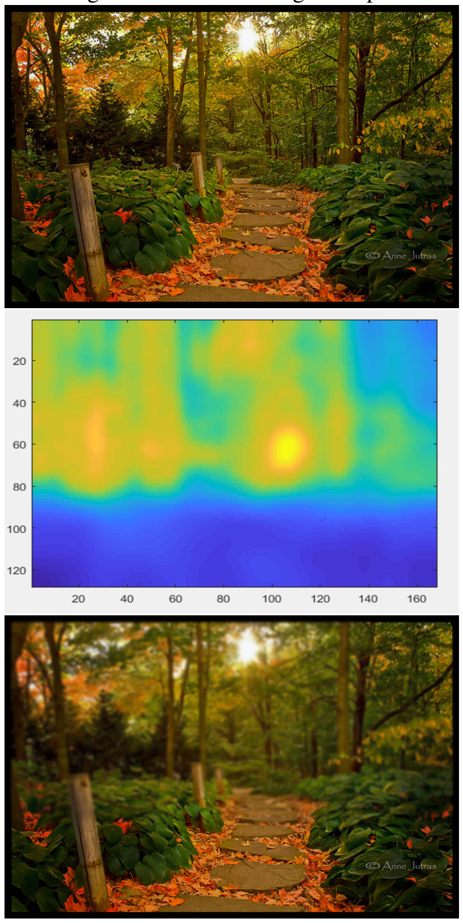
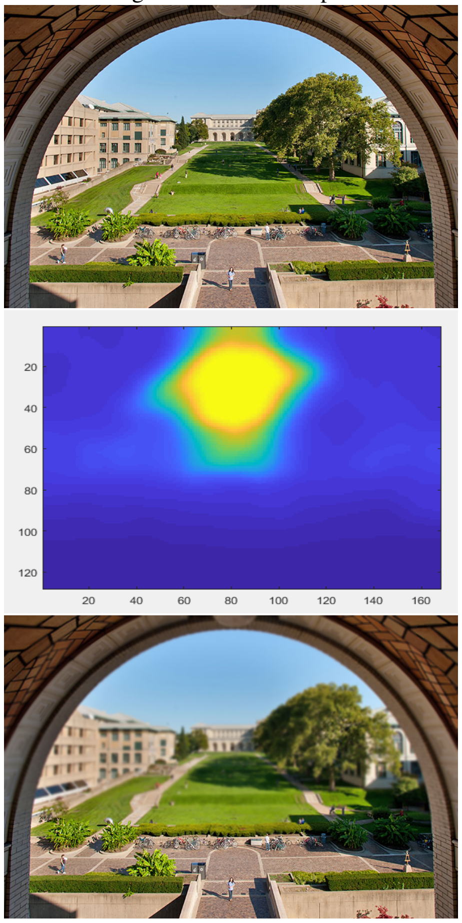


Figure 4. CMU Example



References

[1] https://jonbarron.info/BarronCVPR2015.pdf

[2] http://www.cs.cornell.edu/ asax-
ena/learningdepth/NIPS_LearningDepth.pdf