Variable Synthetic Depth of Field with Mobile Stereo Cameras

Eric Yang
eyyang@stanford.edu

Abstract

We implement a pipeline to perform variable depth of field simulation for mobile devices based on common camera parameters. Overall, our pipeline given good depth information works well at adjusting focal distance, aperture size, and bokeh shape with minor artifacts. Though our local-block disparity matching algorithm performs poorly on the KITTI stereo challenge with 68% error, it is decent enough to estimate depth for synthetic depth of field in large baseline, well-calibrated cameras. Estimating this information from a mobile smartphone camera proves to be a challenge for our disparity method due to differing lens parameters, asynchronous shots, and small baseline.

1. Introduction

Depth of field is the depth range at which a scene is sharp or in focus, while parts outside of the range are blurred due to defocus. At far depths, light sources are heavily blurred by the shape of the camera's aperture, producing blurred lights of such shape in the final image called bokeh. The depth of field and bokeh effects are thus desirable in high-quality photographs as these are traditionally captured on expensive cameras with large apertures and long focal lengths.

Therefore, it is desirable to achieve these appealing effects on cheaper, smaller mobile cameras using computational methods to post-process all-in-focus images. Furthermore, it is important to allow flexibility in the parameters used for this process to adjust focus and intensity of this effect, just as a professional photographer would with a traditional DSLR.

In this work, we simulate the depth of field on images using simple camera parameters to provide variable post-processed depth of field effects for mobile photography. We resolve depth information from a scene using stereo disparity matching.

Our contributions include:

- Designing a depth of field simulation pipeline with tunable camera parameters (focal length, focus plane, aperture size, and aperture shape) for mobile images.
- Implementing and evaluating a local-block stereo disparity matching algorithm to estimate depth for our pipeline.

2. Related Work

A notable work by Google implemented synthetic depth of field for portrait photos [6]. They used a single camera with dual pixels to compute blur kernels directly from disparity. The subjects were designed to be people only, as a neural network was trained to segment people. Their algorithm was tuned for mobile phone performance and efficiency. Another method by DeepLens [7] used purely neural networks to both segment subjects and estimate their depths. The goal of these were aimed to mainly be a point-and-shoot method without flexible camera parameters that we aim to accomplish. They also focus on specific targets to segment based on training data, but we believe this is not necessary nor generalizable if we can achieve good enough depth maps.

We choose to use depth from stereo vision as it appears the most general and accessible in mobile photography. Training a neural network to estimate monocular depth is a hard challenge that requires learning complicated semantics and cannot be robust and general enough for the variety of images taken. Depth sensing cameras such as RGBD are not nearly as common as multiple cameras on mobile devices. Traditional stereo matching algorithms use cost metrics and smoothing to obtain disparity for depth estimation [5]. Google's portrait mode estimates it similarly with their dual pixel cameras, while refining their disparity map using a bilateral solver to maintain details within edges of a reference RGB image [1]. Other work on stereo disparity matching on mobile devices include using neural networks like AnyNet [8].

3. Method

Our synthetic depth of field pipeline is shown in Figure 1. The intensity of blur as a function of distance is parameterized by camera parameters shown in the circle of confusion Equation 3. The size of blur is discretized to kernel radii in pixel counts. The input image is blurred by each kernel and

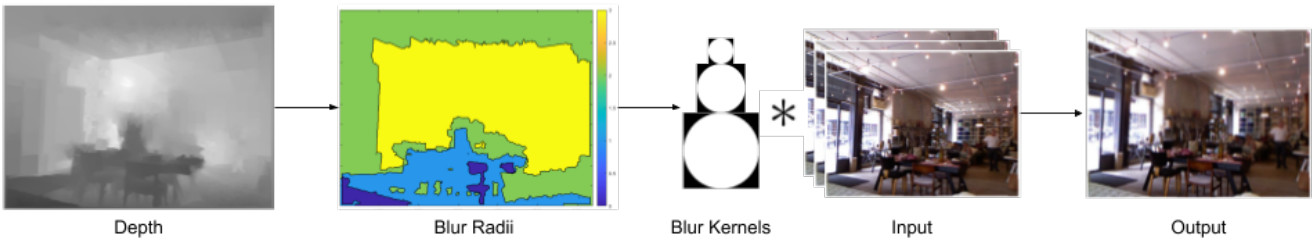

Figure 1. Our pipeline uses a depth map and input image to create map of blur radii. With each blur kernel of a radius, the input is blurred. The output is a masked sum of all of the blurred images.

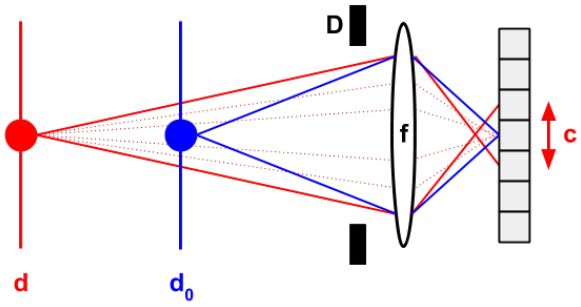

Figure 2. Diagram showing the parameters for the circle of confusion. The red point is out of focus, with its light spreading to multiple pixels on the sensor at a non-zero size c, where the focused blue point is not blurred. A larger aperture D permits larger angles of light to create a larger c.

combined by masking each distinct blurred region.

3.1. Circle of Confusion

The circle of confusion formula is based on the geometry of Figure 2 describing the size of the circle c that a point at distance d from the lens creates on the sensor as a function of the focal plane d, focal length f, and aperture diameter D. These parameters are what we control for the defocus blur.

$$c(d) = D(\frac{f}{d_0 - f})(\frac{|(d - d_0)|}{d}) \quad (1)$$

The circle of confusion is converted from absolute size to a map of radii r in unit pixels for each pixel i using the equation below, where p is the pixel size. Since for arbitrary images the pixel size of the sensor is unknown, the pixel size is estimated by scaling a reference camera's pixel size p_0 by the ratio between the reference's sensor resolution in pixels H_0 with the input image's height in pixels H. We use $p_0 = 7.5\mu m$ and $H_0 = 4000$ for a 24 megapixel camera.

Figure 3. Visualization of blur kernel shape, where white areas are non-zero weights. The weights in kernel are uniform and normalized to one.

$$r_i = \lfloor \frac{c(d_i)}{2p} \rfloor = \lfloor \frac{c(d_i)}{2p_0 \frac{H_0}{H}} \rfloor \quad (2)$$

3.2. Blur Kernels

For each radius in the radii map, we construct a blur kernel that describes the point spread function (PSF) of the defocus. The blur kernel $B_r \in R^{(2r+1)\times(2r+1)}$ corresponds to the spreading of the point light in the scene to an area on the camera sensor. This area is affected by the size and shape of the aperture. While the aperture diameter D is accounted for in the prior circle of confusion equation, the aperture shape is selected at this stage from a library of shape-forming piecewise functions. We defined three shapes shown in Figure 3. Though the physical PSF is for a circular aperture is the Airy disk, we approximate it using a shape with uniform weights.

3.3. Depth from Stereo Matching

Our stereo matching algorithm uses rectified stereo images to match pixels. Rectification aligns the stereo images such that the disparity for one pixel in an image can be searched for along the same row in the other image. We use the left image as the input image and the right image as a reference for disparity. We follow a simple local block-matching algorithm to compute the cost of each disparity

and cost aggregation to smooth the costs before selecting the best disparity per pixel.

3.3.1 Match Cost

For each pixel in the left image $L_{(x,y)}$, we compute the cost to match it with another pixel in the right image $R_{(x-\delta,y)}$ with the horizontal pixel disparity δ. We subtract the disparity because the right image will always show the scene shifted to the left. The costs are stored in a cost volume $C \in R^{X \times Y \times (\delta_{max}+1))}$, where X and Y the image size and each channel of C represents the costs at each disparity.

The cost at each pixel is the difference between the block or neighborhood around $L_{(x,y)}$ and $R_{(x-\delta,y)}$. The neighborhood is represented as a 7x7 square window around the pixel $W(\cdot)$. We use the common squared sum of differences (SSD) between the blocks.

$$C(x, y, \delta) = \sum_{W} (W(L_{(x,y)}) - W(R_{(x-\delta,y)}))^2 \quad (3)$$

3.3.2 Cost Aggregation

The cost aggregation step applies a filter to each disparity layer of the cost volume to smooth the costs in the spatial dimension. This greatly reduces the noise in the final disparity map and helps the disparities to agree with neighboring disparities. We use a simple 7x7 box filter, a square filter with uniform weights that sum to one. After cost aggregation, the disparity is the argmin of the cost volume along the disparity dimension.

3.3.3 Depth Calculation

With disparity, the depth can be calculated with the following equation. The baseline is the distance between the two cameras, f_{cam} is the focal length of the camera in unit pixels, and δ is the disparity in pixels.

$$d = baseline \times \frac{f_{cam}}{\delta} \quad (4)$$

4. Results & Analysis

We perform our analysis on three different types of data:

1. NYU V2 dataset: ground truth depth maps.
2. KITTI stereo dataset: calibrated stereo images.
3. Mobile camera: self-collected images from two different cameras on an iPhone.

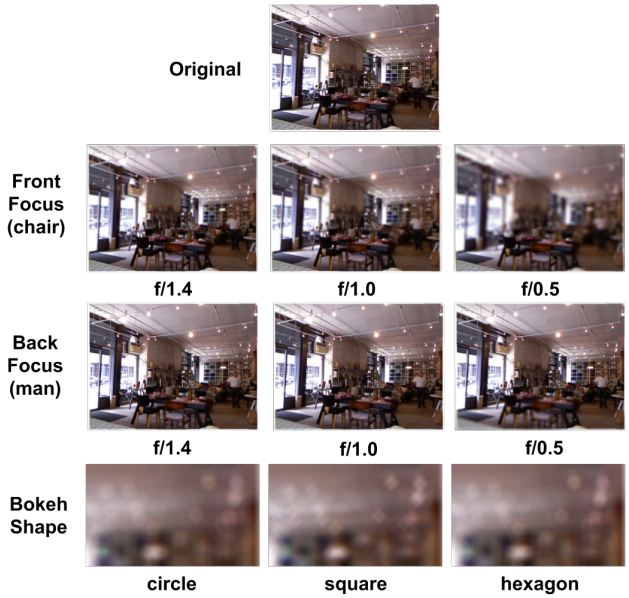

Figure 4. Our pipeline can focus on different distances from the chair in front to the man in the back. We can change the blur intensity by changing the aperture size. We achieve interesting bokeh effects by changing the kernel shape.

4.1. Ground Truth Pipeline

We first test our pipeline using a ground truth depth map from NYU V2 [4], which were taken with Microsoft Kinect cameras and processed to fill in missing data. A demonstration of the types of outputs we are able to achieve are shown in Figure 4. We are able to achieve realistic depth of field effects at various focal planes and adjust the aperture to change the intensity of blur. Additionally, we are able to create bokeh of the shaped defined by the blur kernel shapes. As we do not have a reference photo taken from a DSLR, we cannot compare the exact accuracy of the blur corresponding to each f-number.

We notice artifacts resulting from our method of blur and mask. The edges of one blurred and depth-masked image bleeds into neighboring layers as seen in Figure 5. This is to due the edge first being blurred and masked at the original edge location.

4.2. KITTI Stereo

The KITTI driving dataset [3] includes rectified and calibrated images from its stereo rig. The resulting depth map and images from our stereo algorithm and depth of field pipeline are shown in Figure 6. We evaluate our disparity matching on the KITTI stereo training set of 200 pairs of images. The evaluation criteria is the percentage of erroneous disparity pixels, defined by a disparity that is greater

Figure 5. Our blurring and masking method causes bleeding artifacts over edges.

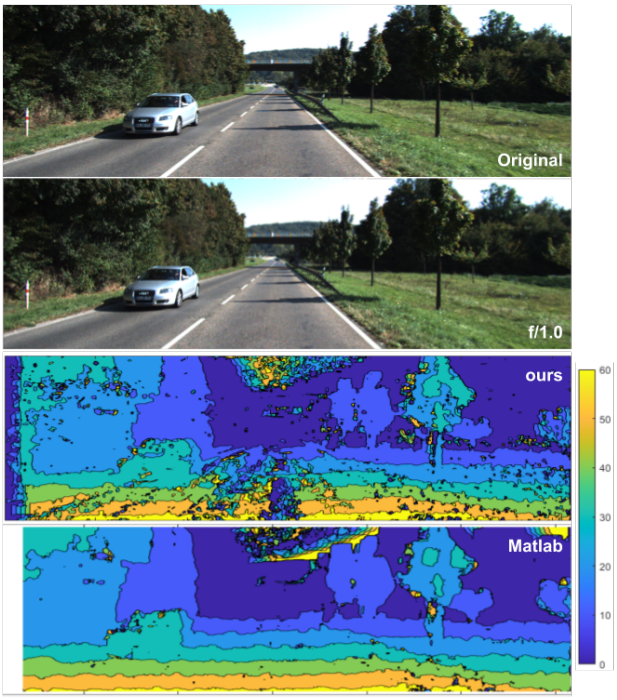

Figure 6. Here we focus on the car in the KITTI dataset. The background is blurred by up to three radii kernels. Even though our disparity map is more noisy, we the resulting image is not affected with major artifacts.

than3 pixels and 5% from the ground truth. We compare our results to Matlab's disparityBM function and the state-of-the-art method Convolutional Spatial Propagation Network (CSPN) [2] in Table 1. We point our that our matching error is much higher than the other methods, as we can see that our disparity maps are much noisier and less smooth.

	Ours	Matlab	CSPN
Error	68.25%	13.59%	1.74%

Table 1. Our disparity matching method compared to Matlab disparityBM and the state-of-the-art on the KITTI challenge. Ours and Matlab were evaluated on the training set, while CSPN was based on the test set leaderboard on the KITTI website.

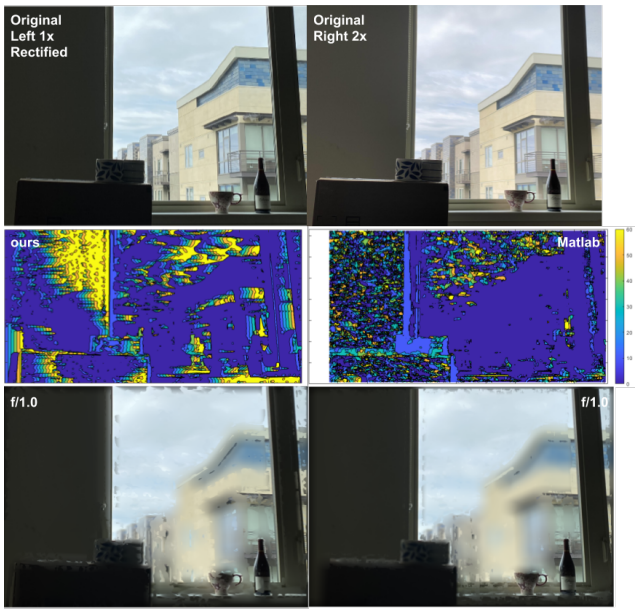

Figure 7. Both our matching algorithm and Matlab's fail to create good enough depth map to blur the images accurately. The result is zero disparity in many locations which corresponds to infinite depth and a large blur kernel. We see that the lighting and sharpness of the two images are slightly different.

4.3. iPhone Stereo

The final test for our implementation is for a real mobile phone. We used the iPhone 11 Pro with a 1x zoom wideangle camera and a 2x zoom telephoto camera. We collected our own checkerboard images to calibrate the cameras with Matlab's stereoCameraCalibrator application and rectify with rectifyStereoImages. Since we had no means to take these photos simultaneously, we took the stereo photos sequentially by manually switching between the modes and keeping the device static.

Our results are shown in Figure 7. Both our disparity matching and Matlab's disparityBM were unable to create suitable disparity maps to produce a good image. The artifacts are produced due to failure to match pixels correctly. This is most likely due to slight errors in calibration and rectification and the extra complexity that the two cameras do not use the same lens and are taken at asynchronous times, so the distortion effects and white balancing may differ to

cause the matching to fail. We also note the the images were received in a JPEG compressed format, which would further misguide the matching. It may also be that since the baseline and pixels are very small, the minute disparities are even harder to detect and result in zero in many cases.

5. Conclusion

We have shown that our implementation of a variable depth of field simulation pipeline for mobile cameras works decently under more ideal conditions, while it needs further improving in actual mobile device configurations.

Our pipeline has strengths in using a more physically accurate circle of confusion simulation as opposed to Google's linearly increasing blur radius with disparity. [6]. It also allows focusing on different object depths and bokeh shaping. It's weakness is in merging blurred images, as our simplistic method causes artifacts near edges. A segmentation method would help to remove foreground objects from the background when blurring to eliminate bleeding. Another improvement to make is constructing a more physically accurate PSF and evaluating if it improves the quality of blur.

The main weakness of our project is the depth estimation. While our stereo matching algorithm created a decent enough disparity map to apply depth of field on KITTI stereo images, it was not applicable to our mobile images. Though disparity matching may do better if we could clean up the iPhone images to be more consistent, it would require more perfect calibration and matching white balance. Our stereo algorithm could improve as well, as the overall disparity error is much higher than the what we compared with. Overall, this shows the challenge of working with different kinds of lenses and small stereo baselines in multi-camera mobile devices. It also shows that much more is needed to create perfect disparity maps, such as bilateral smoothing [1] and deep learning techniques [2].

References

[1] J. T. Barron and B. Poole. The fast bilateral solver. In *ECCV*, 2016.

[2] X. Cheng, P. Wang, and R. Yang. Learning depth with convolutional spatial propagation network. *IEEE Transactions on Pattern Analysis and Machine Intelligence(T-PAMI)*, pages 1–1, 2019.

[3] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[4] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[5] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47:7–42, 04 2002.

[6] N. Wadhwa, R. Garg, D. E. Jacobs, B. E. Feldman, N. Kanazawa, R. Carroll, Y. Movshovitz-Attias, J. T. Barron, Y. Pritch, and M. Levoy. Synthetic depth-of-field with a single-camera mobile phone. *CoRR*, abs/1806.04171, 2018.

[7] L. Wang, X. Shen, J. Zhang, O. Wang, Z. Lin, C. Hsieh, S. Kong, and H. Lu. Deeplens: Shallow depth of field from A single image. *CoRR*, abs/1810.08100, 2018.

[8] Y. Wang, Z. Lai, G. Huang, B. H. Wang, L. van der Maaten, M. Campbell, and K. Q. Weinberger. Anytime stereo image depth estimation on mobile devices. *arXiv preprint arXiv:1810.11408*, 2018.