Bokeh-licious: Shallow Depth of Field for Images using Portrait Segmentation and Depth Estimation

Shreyas Nikam
Northeastern University
nikam.sh@northeastern.edu

Rohit Awate
Northeastern University
awate.r@northeastern.edu

Abstract

The paper presents a method for using portrait segmentation and depth estimation from monocular images to create a shallow depth of field in images, resulting in a bokeh effect. This is widely known as *portrait mode* in modern smartphone cameras. This technique allows for more artistic control over the focus of an image, producing a more pleasing visual aesthetic. A combination of deep learning techniques and image processing is applied to recreate images with a progressive bokeh. The method is demonstrated through experimentation and comparison to traditional techniques to recreate the effect using simpler model architectures. It emulates the effect of a large aperture and/or focal length from bigger cameras and lenses while blurring the background elements for aesthetic value as well as better separation between the subject and the background. We propose an approach that uses deep learning to perform segmentation for isolating the subject and a depth estimation model on the background elements to understand the relative distance between them which is then used to apply a progressively stronger Gaussian blur.

1 Introduction

The use of shallow depth of field in photography can create visually appealing images by drawing the viewer's attention to a particular subject. However, achieving a shallow depth of field can be challenging, especially when shooting with a smartphone camera. In this paper, we propose a novel approach for creating shallow depth of field in images using portrait segmentation and depth estimation.

Our approach first uses a deep learning model to perform portrait segmentation, which allows us to identify the subject of the image. We then use another deep learning model to estimate the depth within the image, which provides us with information about the relative distances of different objects in the scene. Based on this depth information, we apply a progressive Gaussian blur to the background of the image, effectively simulating a shallow depth of field which can be better visualized in 1.

We evaluate our approach on a variety of images and show that it can produce visually appealing results. In addition, we compare our approach to another method for achieving shallow depth of field and show a comparative analysis in terms of visual quality and computational efficiency. Our approach can be easily integrated into existing image editing software and offers a simple yet effective way to create a shallow depth of field in images.

1.1 Open Source Implementation References

We compare our models against existing implementations and use the open-source code implementations available for Portrait Segmentation [1] and Depth Estimation [2]. Apart from these, we make

[1] https://github.com/ternaus/people_segmentation
[2] https://github.com/ialhashim/DenseDepth.git

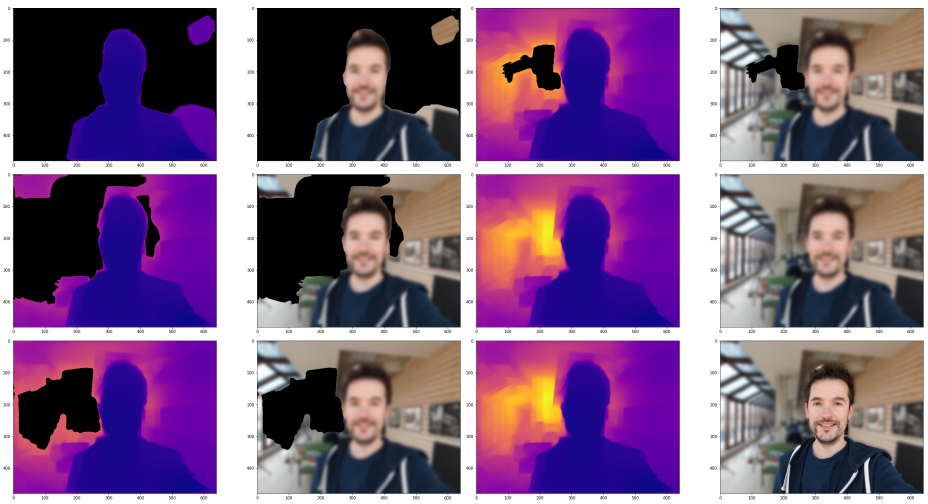

Figure 1: Visualization for the progressive blur using depth information superimposed by the portrait generated from the portrait mask. The full-sized image can be viewed here

.

use of several open-source libraries, such as PyTorch, NumPy, OpenCV and more. These libraries provide us with valuable tools and functionality that greatly enhance our work. We are grateful for the hard work and dedication of the developers and maintainers of these libraries, which have made our research possible.

1.2 Contributions

The task consisted of two functional parts, viz. Portrait Segmentation and Depth Estimation. The work was equally divided among the two members and the following section describes the contributions of each individual:

1. Rohit was responsible for portrait segmentation, not limited to but including the study of the various architectures on the specified task, ablation study, generating alpha matte, and distinguishing between the background and foreground layers.
2. Shreyas carried out the study of depth estimation from monocular images and generated these depths, post which both individuals focused on portrait segmentation. Binning was performed on the depths, to cluster objects belonging to the same depths and applied a progressive depthwise Gaussian blur.
3. Rohit superimposed the foreground and background images to generate the final bokeh licious image, consisting of the progressive blur and the sharp in-focus person matte.
4. Shreyas created a prediction pipeline using baseline depth prediction and portrait segmentation models to compare the results of the two approaches.
5. The documentation, code contribution, report, and presentation-related tasks were performed together, and have an equal weightage in the contributions of each person.
6. We release all the data, code, and trials and experiments related to the project on GitHub [3].

1.3 Limitations and Assumptions

The method will only work on images that contain only humans because the portrait segmentation algorithm is designed to work specifically on images containing human subjects. If the image contains other objects or scenes, the algorithms may not be able to accurately segment the portrait and estimate the depth information, which would affect the quality of the shallow depth of field effect. Results show that the algorithm treats multiple humans as seen in 2.

[3] `https://github.com/RohitAwate/Bokehlicious`

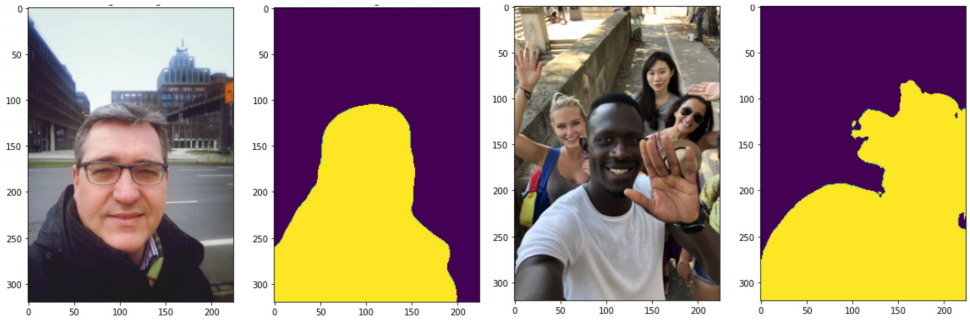

Figure 2: Fail case of portrait segmentation in case of multiple humans.

The resolution of the input image may affect the accuracy of the portrait segmentation and depth estimation algorithms. Lower-resolution images may not contain enough information, which could lead to less accurate segmentation and depth estimation, and consequently, a lower-quality bokeh.

The shallow depth of field effect will not be able to accurately reproduce the exact depth of field of the original scene, but only approximate it based on the estimated depth information.

2 Related Work

The application of the bokeh effect is discussed in the study utilizing an encoder-decoder architecture, in which the picture is processed for salient region segmentation and depth estimation using dedicated pre-trained encoders and two decoders to carry out these tasks independently. The picture is then produced using the intensity values from the decoder's output, which is responsible for determining depth through another devoted CNN architecture. We plan to put together a similar approach through our project, thus, establishing (1) as our base paper.

The research in (2) demonstrates an analogous technique to that described in (1) for mobile devices utilizing the MobileNet architecture, which performs admirably when inference time and memory limitations are considered. The approach in (3) describes bokeh effect rendering leveraging a deep PyNet CNN architecture for automatically identifying spatial information. The major contribution of this paper is the EBB dataset consisting of 5K shallow and wide depth-of-field image pairs. (4) explores Structure from Motion (SfM) and Multi-View Stereo (MVS) images over the internet and combines them to form a large dataset for presenting a single view depth prediction system using machine learning. (6) presents a comparative analysis of the different loss functions used for depth estimation from single monocular images to achieve state-of-the-art results. We refer to this paper to try out different variations of the loss functions and their combinations while training our model.

Finally, a novel approach is devised in (5), which estimates depth from visual cues in the image such as edges, hue values, saturation, color information, and depth priors. This study presents a computationally inexpensive algorithm as compared to deep learning models with almost comparable results to the ground truth.

3 Methods

3.1 Overview

Influenced by the implementation described in (1), we follow a similar approach, in which we utilize two U-Net architectures for generating depths and portrait mattes. The overview of the architecture can be seen in 3.

3.2 Datasets

We employ two different datasets for the two tasks described.

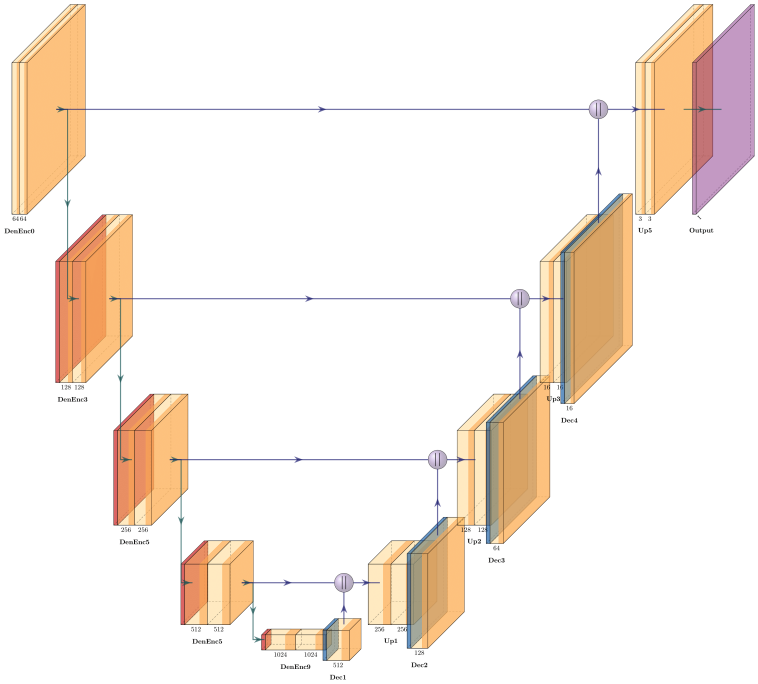

Figure 3: Model Architecture. The full-sized image can be viewed here
.

1. For the task of portrait segmentation, we use the PFCN (7) dataset which consists of 1800 images containing 600x800 self-portraits and their masks collected from Flickr. The dataset contains 1500 training images and 300 test images.
2. The NYU Depth dataset (9) is a large-scale dataset of RGB-D images, containing more than 1449 densely labeled pairs of aligned RGB and depth images. It was collected by the NYU Machine Perception and Robotics Laboratory and features a wide variety of scenes, including indoor and outdoor environments, with a diverse set of objects and surfaces.

3.3 U-Net Architecture

The U-Net architecture is a deep-learning model widely used in medical image analysis and has achieved state-of-the-art performance on many tasks. It is known for its ability to effectively learn from small datasets and perform well on images with high levels of noise and variation.

In the U-Net model, the encoder part of the network downsamples the input image, reducing its spatial resolution and increasing the number of channels. This is done by applying convolutional filters and max-pooling layers.

The decoder part of the network then uses upsampling layers to increase the spatial resolution of the feature maps produced by the encoder. We use bilinear interpolation for upsampling. Bilinear interpolation is a method of upsampling in which the new pixel values are calculated by taking the weighted average of the surrounding pixels in the input image. In addition to this, we also employed batch normalization at each decoding layer.

One key feature of the U-Net architecture is its use of skip connections. Skip connections are used to concatenate the feature maps from the encoder with the upsampled feature maps from the decoder. This helps to incorporate rich, high-level, spatial information from the encoder into the low-level information produced by the decoder.

Initially, we used a vanilla U-Net architecture, but then switched to using pre-trained weights of the DenseNet-121 architecture in our encoder and creating skip connections from layers numbered 11, 9, 7, 5, 3, and 0 to the decoder respectively in that order. This allows the network to retain fine-grained details in the output. More information on this can be found in section 4.2.1.

Model	Test Loss	Test Accuracy
U-Net (baseline)	0.921	0.863
U-Net + DenseNet encoder	0.578	0.927
U-Net + batch norm	0.837	0.888
U-Net + upsampling	0.901	0.989
U-Net + DenseNet encoder + Post-processing	**0.561**	**0.993**

Table 1: Evaluation metrics for different versions of the U-Net architecture for portrait segmentation.

Model	Test Loss	Test Accuracy
U-Net (baseline, pre-trained)	**0.601**	**0.912**
U-Net + DenseNet encoder	0.932	0.527
U-Net + batch norm	0.837	0.588
U-Net + upsampling	0.901	0.589
U-Net + DenseNet encoder + Post-processing	0.961	0.693

Table 2: Evaluation metrics for different versions of the U-Net architecture for depth estimation.

3.4 High-level algorithm

1. Load input image
2. Compute its edge map using Sobel filter and append it as a 4th channel
3. Pass this image to the portrait segmentation model to generate the portrait mask
4. Pass the same image to the depth estimation model and generate the depth map
5. Discretize the depth information into buckets such as $0 - 0.2$, $0.2 - 0.4$ and so on.
6. Apply a progressively stronger Gaussian blur to these buckets. This would result in the first bucket i.e. one containing pixels with depth in range $0 - 0.2$ to be blurred less than the next bucket containing the pixels from the depth in range $0.2 - 0.4$ and so on.
7. Stack the pixels from all buckets together to form a single image.
8. Superimpose the portrait mask onto the previous stacked output resulting in the final image with a rich bokeh.

4 Experiments

4.1 Ablation Study

An ablation study is a type of experiment that is used to evaluate the importance of individual components or techniques in a complex system. In the case of the U-Net architecture, we evaluated the impact of using a DenseNet encoder, batch normalization, skip connections, and upsampling on the performance of the network. We trained multiple versions of the U-Net architecture, each with a different combination of components or techniques. For example, we trained one version with a DenseNet encoder and skip connections, another version with a DenseNet encoder and upsampling, another version with batch normalization and skip connections, and so on. Once the networks were trained, we evaluated their performance on a test set and compared the results. This allowed us to determine the relative importance of each component or technique in the overall performance of the network. A comparative analysis is shown in 1 and 2

4.2 Model Design and Training

We used a mostly similar U-Net architecture for both our models. Details in following sub-sections.

4.2.1 Portrait Segmentation

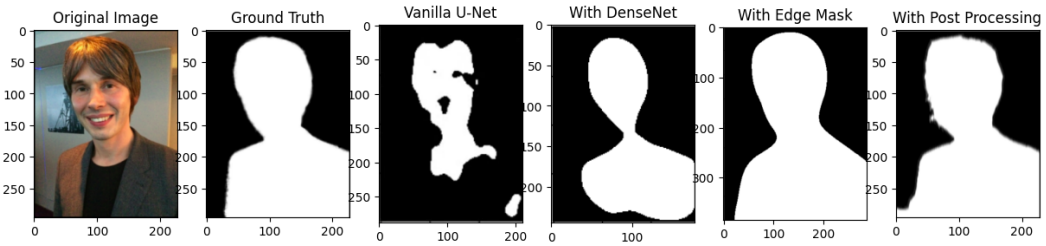

Figure 4: Output masks across different architectures we experimented with.

Portrait segmentation at its core is a binary classification task that runs per pixel producing 0 or 1 depending on whether the pixel is part of the background or foreground. U-Net lends itself perfectly to this task.

Our initial design was a vanilla U-Net architecture. The segmentation results can be seen in 4. It's apparent that the model was able to learn, but would need more epochs. We quickly realized that training an effective model from scratch would be computationally too intensive and take several hours.

At this point, we decided to use a pre-trained model as our encoder. We chose DenseNet for the job. We carefully observed its architecture and determined from which layers we can extend skip connections to wire up to our decoder, which also needed a few changes. As is evident from 4, this considerably improved the performance of the model. We tried training this for roughly 40-50 epochs but didn't see any improvement in the output.

Next, we devised a novel approach to feed additional data to the network. It can be seen from the output of the previous trial that the model's output is not quite *filling up* the region of the person in the image. Our intuition was to first perform edge detection on the input image and add the edge mask as an additional channel to the model's input. We use this channel in the last stage of the decoder where we convolve it with the output from the previous layers of the decoder to produce the final output. This gave stellar results as observed in 4.

With the edge mask, the output was reasonably close in shape to the ground truth. However, the edges lacked detail. Training with additional epochs didn't lead to favorable results. We decided to experiment with some image processing at this point. We convolved the model's output with a max filter which augmented its edges. We then subtracted the original output from it which gave us just the area around the border. This can be seen in 5. Next, we used this as a mask to only grab the edge detail from the detection output. We applied a Gaussian blur on this to smooth out some of the finer edge details and combined this with the original model output to produce the final portrait mask that can be seen in 4. This preserves a lot of the edge detail.

The confusion matrix 6 further shows that our false positive and false negative rates are orders of magnitude lower than our true positives and true negatives. Note that this confusion matrix is computed across the entire training set.

Figure 5: Border mask

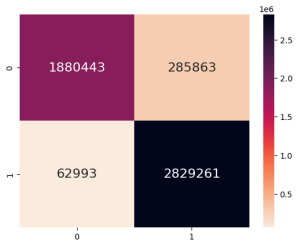

Figure 6: Confusion Matrix

4.2.2 Depth Estimation

Similar to portrait segmentation, we attempted to achieve depth estimation with a U-Net architecture and a DenseNet encoder. However, the specified task turned out to be difficult for the model to achieve in the limited resources and time frame available. The outputs of the same can be seen in7. The outputs seemed to learn about the intensity values of the input instead of estimating the depth. We intuit the problem lies with the amount of data we chose for training the model in a limited resource environment (Google Colab). Studies have mentioned that they were required to train the model for at least 24 hours on multiple GPUs, which was not readily available to us.

We made an effort to achieve an accuracy of levels similar to the same using various ablation studies mentioned in 2. However, they seemed to not reach optimum levels of output regardless. We compared the results of the outputs against the pre-trained model and finally resorted to using the pre-trained model to employ the progressive blur based on depth in our final processing pipeline.

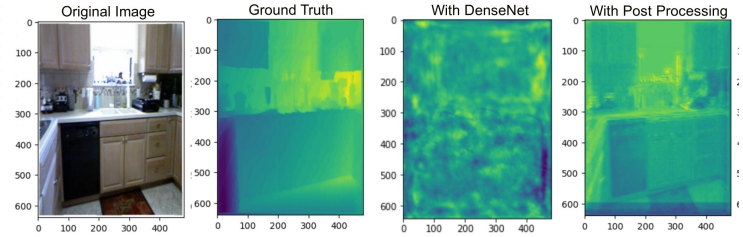

Figure 7: Depth estimates across the architectures

4.3 Hyperparameter Tuning

We tried a variety of hyperparameters before settling for the final figures mentioned below.

Initially, we experimented with a relatively conservative learning rate of 1e-3 which led to decent results, but not much decline in loss. We realized that the model was underfitting. We tried a learning rate of 1e-2 which led to a modest increase in output quality.

We also experimented with a variety of different loss functions such as Binary Cross Entropy with Logits, Cross Entropy, Mean Squared Error, Dice, SSIM and L1. Much to our surprise, L1 led to the best output.

We also measured the impact of using different batch sizes on the training time. We found that using a larger batch size (e.g. 128) resulted in faster training times, but also required more GPU memory. This trade-off must be considered when optimizing the training process for a given hardware setup.

Finally, we decided to use a far more aggressive learning rate and pair it with a LR scheduler that is available with PyTorch. We chose ReduceLROnPlateau and set it to be quite aggressive as detailed below. This led to a dramatic increase in output quality.

Our final configuration was as follows:

1. Loss function: L1

2. Optimizer: SGD
3. Learning Rate Scheduler: ReduceLROnPlateau (patience=0, factor=0.9)
4. Learning Rate: 0.1
5. Momentum: 0.99
6. Batch size: 8
7. Epochs: 30

4.4 Training Environment

We trained our models on Google Colab. This allowed us to take advantage of the parallel computing capabilities of the their CUDA-enabled NVIDIA GPUs, which significantly reduced the training time compared to using a CPU-only machine. Overall, our results demonstrate the effectiveness of using a CUDA-enabled GPU for training deep-learning models on vision workloads and highlight the importance of considering hardware choices when optimizing training times.

5 Conclusion

5.1 Summary

In conclusion, shallow depth of field can be achieved in portrait photography by using depth estimation and portrait segmentation techniques to identify the areas of the image that should be in focus and those that should be blurred. By applying a progressive blur to the image based on the estimated depth of each pixel, it is possible to create the illusion of shallow depth of field and draw the viewer's attention to the subject of the photograph. This technique can be useful for creating more dynamic and engaging portrait images.

5.2 Inferences

We were able to successfully outperform baseline implementation for portrait segmentation, however, depth estimation is a comparatively challenging task that requires more fine-tuning and a lot of training. An interesting observation during our study was increasing batch size for a fixed learning rate had the same effect as decreasing learning rate for a fixed batch size by the same factor.

Applying a progressive blur based on the depth of an object can create a more natural and realistic-looking shallow depth of field effect. Batch normalization plays a significant role to help improve the stability and convergence of neural networks during training. This is because batch normalization normalizes the inputs to each layer of the network, which can help prevent the internal covariate shift that can occur when the distribution of the inputs to each layer changes during training. This in turn can help the network learn more efficiently and effectively, leading to improved performance on tasks such as image classification or machine translation.

Using a pre-trained model in our encoder achieved better performance than a model that is trained from scratch on the same task because the pre-trained model had already learned many of the important features and patterns that are relevant to that task. Additionally, training a model from scratch can be computationally expensive and time-consuming, while fine-tuning a pre-trained model is generally faster and more efficient. Using a correct loss function is important for guiding the learning process for the specified tasks. If an incorrect loss function is used, the network may not learn effectively and may not be able to solve the task it is designed for. We were able to witness these effects during our study.

5.3 Future Work

For our future scope, we plan to employ state-of-the-art GANs and stable diffusion architectures for recreating images with progressive blur. We can also contribute a dataset of the outputs of our model since no datasets are available that employ shadow depth of field effect to images containing self-portraits.

References

[1] K. Purohit, M. Suin, P. Kandula and R. Ambasamudram, "Depth-Guided Dense Dynamic Filtering Network for Bokeh Effect Rendering," 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), 2019, pp. 3417-3426, doi: 10.1109/ICCVW.2019.00424.

[2] Sergio Orts Escolano and Jana Ehman, "Accurate Alpha Matting for Portrait Mode Selfies on Pixel 6", Google AI Blog, https://ai.googleblog.com/2022/01/accurate-alpha-matting-for-portrait.html

[3] Ignatov, A., Patel, J., and Timofte, R., "Rendering Natural Camera Bokeh Effect with Deep Learning", arXiv e-prints, 2020.

[4] Li, Z., & Snavely, N. (2018). MegaDepth: Learning Single-View Depth Prediction from Internet Photos. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2041-2050.

[5] T. -Y. Kuo, Y. -C. Lo and Y. -Y. Lai, "Depth estimation from a monocular outdoor image," 2011 IEEE International Conference on Consumer Electronics (ICCE), 2011, pp. 161-162, doi: 10.1109/ICCE.2011.5722517.

[6] Marcela Carvalho, Bertrand Le Saux, Pauline Trouvé-Peloux, Andrés Almansa, Fréderic Champagnat. On regression losses for deep depth estimation. ICIP 2018, Oct 2018, ATHENES, Greece. ff10.1109/ICIP.2018.8451312ff. ffhal-01925321f

[7] Shen, X., Hertzmann, A., Jia, J., Paris, S., Price, B., Shechtman, E. and Sachs, I. (2016), Automatic Portrait Segmentation for Image Stylization. Computer Graphics Forum, 35: 93-102. https://doi.org/10.1111/cgf.12814

[8] https://www.kaggle.com/code/ajayvamsi123/portrait-segmentation-segnet/data

[9] `https://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html`