Fundamentals of Computer Vision Project Report

Abolfazl Eshagh, Mohammad Yousef Najafi, Sina Imani, Alireza Alipanah, Mohammad Bagher Soltani
Prof. Kasaei Supervisor: Vida Ramezanian Fall 2023

Computational Geometry for Bokeh Effect Generation

Abstract

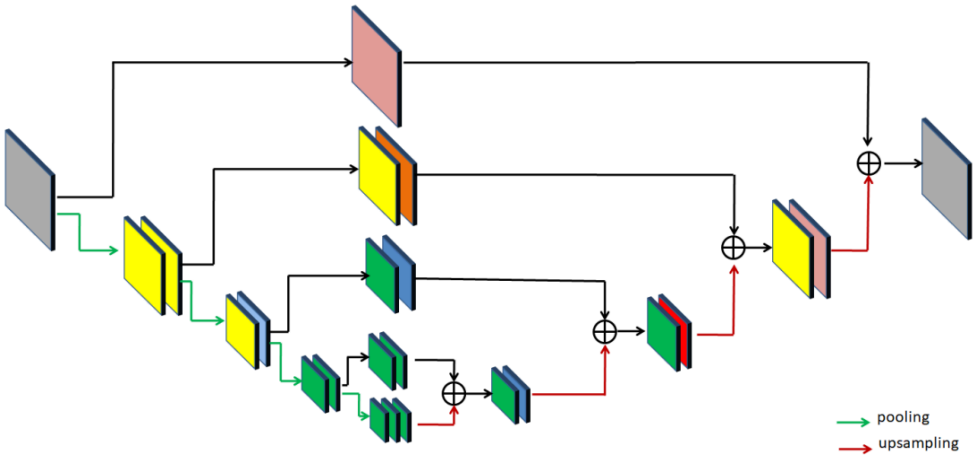
Bokeh effect is used in photography to capture images where the closer objects look sharp and everything else stays out-of-focus, i.e. Bokeh is the shallow-depth of field effect which blurs the background of portrait photos (typically) to bring emphasis to the subject in the foreground. Bokeh photos are generally captured using Single Lens Reflex cameras using shallow depth-of-field. Most of the modern cameras can take bokeh images with the help of a good auto-focus hardware. However, for cameras without a good auto-focus hardware, we have to rely on software to generate bokeh images. This kind of system is also useful to generate bokeh effect in already captured images. Bokeh effect has a wide range of applications, from photography and graphic design to even autonomous cars. First we performed a thorough review fo the previous works done about this problem. There are multiple works using Deep Learning methods to generate Bokeh Effect, but in our project, inspired by a previous work(done by SCPD students) which we explain in the Literature Review section(Paper 4), we use Computational Geometry methods, i.e. **Depth Mapping,** to generate Bokeh Effect.

Literature Review Summary

In the initial phase of our project, we conducted a comprehensive literature review to gather insights and establish a solid foundation for our implementation. Five key papers were thoroughly examined, each contributing unique perspectives and methodologies related to our research focus. These papers, spanning diverse aspects of the field, collectively informed our understanding and shaped the conceptual framework of our project. In the following sections, we provide concise summaries of these works, highlighting their contributions and relevance to our study. The synthesis of these literature sources has been instrumental in guiding our approach and ensuring a well-informed and contextually grounded implementation.

Paper 1: "Depth-aware Blending of Smoothed Images for Bokeh Effect Generation"

The first paper is titled 'Depth-aware Blending of Smoothed Images for Bokeh Effect Generation,' authored by Saikat Duttaa from the Indian Institute of Technology Madras, Chennai, PIN-600036, India. In this paper, an end-to-end deep learning framework is proposed to generate a high-quality bokeh effect from images. The original image and different versions of smoothed images are blended to generate the bokeh effect with the help of a monocular depth estimation network.



Megadepth(Li and Snavely, 2018, shown in the figure above) is used as Monocular Depth estimation network in this work. The encoder part of this network consists of a series of convolutional modules (which is a variant of inception module) and downsampling. In the decoder part, there is a series of convolutional modules and upsampling with skip connections that add back features from higher resolution in between.

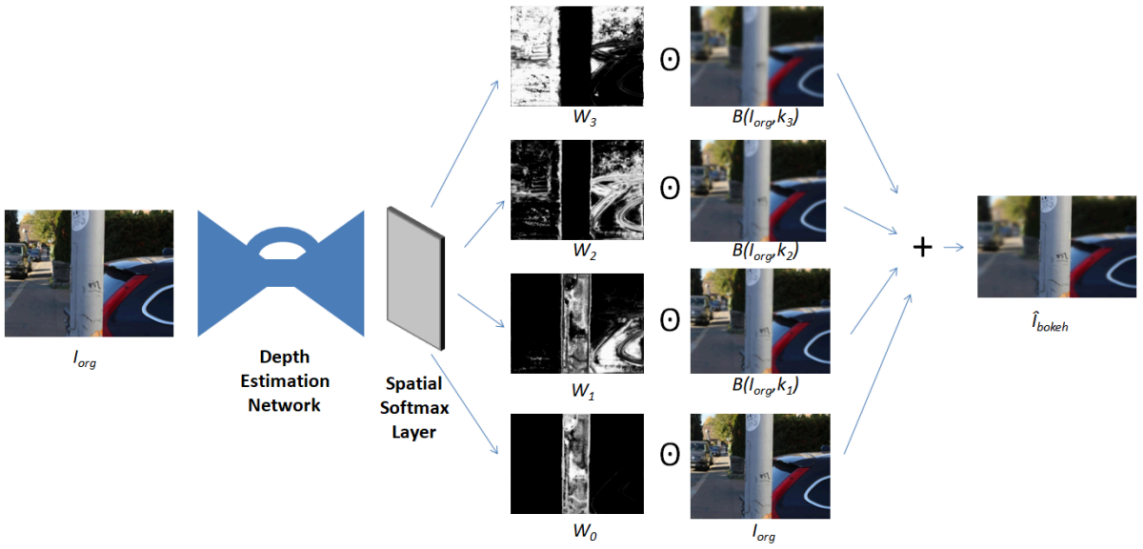
The depth-of-field image can be formulated as a weighted sum of the original image and differently smoothed versions(using different sizes of kernels) of the original image.

$$\hat{I}_{bokeh} = W_0 \odot I_{org} + \sum_{i=1}^{n} W_i \odot B(I_{org}, k_i)$$

where **I_org** is the original image, **B(I_org, k_i)** is image Iorg smoothed by blur kernel of size **k_i ×k_i** and stands for elementwise multiplication, such that for each pixel position **(x, y)**:

$$\sum_{i=0}^{n} W_i[x, y] = 1$$

The weights **W_i** are predicted with the help of a neural network. This can be achieved by modifying the last layer of the pre-trained depth estimation network. Specifically, in the last layer a convolutional layer with 3 × 3 kernel is used along with spatial softmax activation to learn the weights(as shown in the figure below).



Training of the model is done in 3 phases.
In phase 1, the model is trained on smaller resolution images to save training time. Each image from the training set is resized to a resolution of 384 × 512 and the network is trained using **reconstruction loss**. In phase 2, the network is trained using images of resolution 768 × 1024 with reconstruction loss. In phase 3, the model is further fine-tuned with perceptual loss **l_p**, where reconstruction loss and I_p are computed as below:

$$l_r = \left\| \hat{I}_{bokeh} - I_{bokeh} \right\|_1$$

$$l_p = -SSIM(\hat{I}_{bokeh}, I_{bokeh})$$

Paper 2: "[BGGAN: Bokeh-Glass Generative Adversarial Network for Rendering Realistic Bokeh]"

In this paper, the GAN-based method and perceptual loss are combined for rendering a realistic bokeh effect in the stage of fine-tuning the model. Moreover, Instance Normalization(IN) is reimplemented in the network, which ensures the tflite model with IN can be accelerated on smartphone GPU. Experiments show that BGGAN method is able to render a high-quality bokeh effect and process one 1024×1536 pixel image in 1.9 seconds on all smartphone chipsets.
This paper uses EBB! dataset which was released by AIM 2020 Bokeh Effect Rendering Challenge.The dataset pays more attention to synthetic bokeh effect rendering in wide scenes. In this dataset, objects in the depth-of-field area are not only portraits but also other objects, such as road signs, vehicles, flowers, and so on.
The paper takes Glass-Net and Multi-receptive-field Discriminator to construct the BG-GAN. Glass-Net is an end-to-end network that takes an image as an input and produces the result with bokeh effect. Multi-receptive-field Discriminator refines the images generated by Glass-Net to make the final outputs cater to human perception. In addition, TensorFlow Lite frame-work operators are used to reimplement IN to make all of the model operators to compute on smartphone GPU.
Glass Net:

Glass-Net is the novel generator network proposed in the paper for rendering bokeh effects on images. It has an encoder-decoder style convolutional neural network architecture. Glass-Net is a two-stage network. The first stage learns to predict the residual between the input image and ground truth bokeh image. The second stage refines the initial prediction to generate the final bokeh output. Each stage uses 3 downsampling layers in the encoder and 3 upsampling layers in the decoder. Features from the encoder are transformed using 9 residual blocks in the bottleneck. Skip connections carry high-frequency details from encoder to decoder to avoid over-blurring focused regions. The two stages have similar architecture, with more channels in stage 2. Stage 1 has 16/128 channels, while stage 2 has 32/256 channels. Glass-Net is first trained with L1 and SSIM losses to get a reasonable mapping. The multi-receptive field discriminator and perceptual loss then refine the mapping for more realism. In summary, Glass-Net is the core generator network enabling high-quality bokeh rendering using the proposed BGGAN approach.

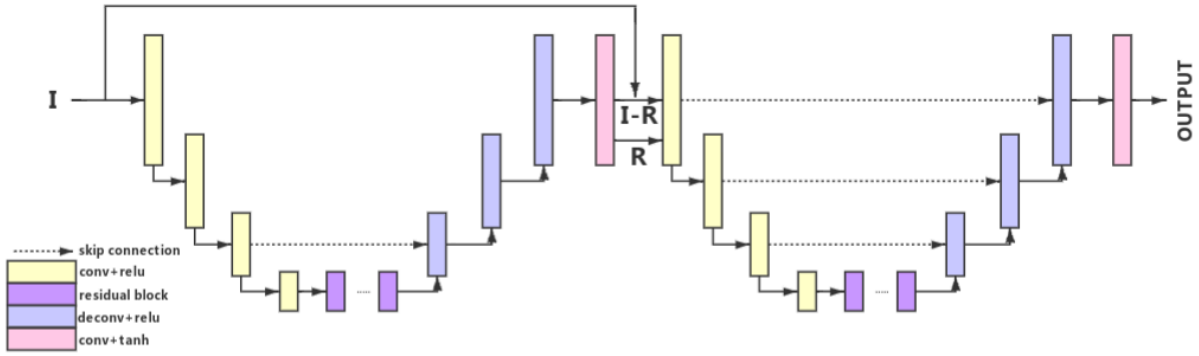


Fig. 1. Demonstration for Glass-Net

Multi-receptive-field Discriminator:
To generate more realistic bokeh images, WGAN-GP is used as a signifi-cant strategy in our network.

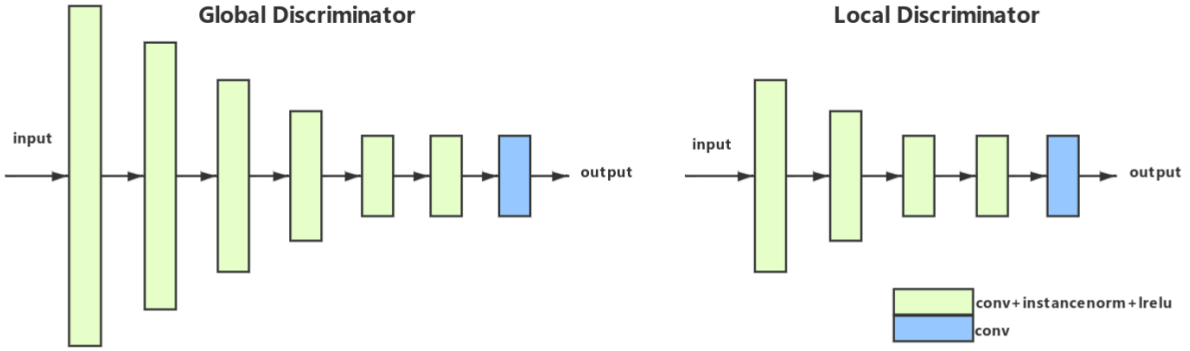


Fig. 2. Demostration for Multi-receptive-field Discriminator

An additional gradient penalty in the loss function BGGAN for Rendering Realistic Bokeh enables the fine-tuning process to be more stable and easier to produce results with higher perceptual quality.
The loss function is also important for fine-tuning the Glass-Net. Aside from L1 reconstruction loss and negative SSIM loss LSSIM , the perceptual loss which computes the Euclidean loss on the feature

maps the relu5 4 layer layer of the VGG19 LV GG and the adversarial loss Ladv for the Glass-Net generator will also be optimized in this stage. The perceptual loss is as follow:

$$L_{VGG} = \frac{1}{HWC} \sum_{i=1}^{H} \sum_{j=1}^{W} \sum_{k=1}^{C} \|F(G(I)_{i,j,k}) - F(C_{i,j,k})\|_1$$

here F (·) denotes feature maps of the 34-th layer of the VGG network which is pre-trained on ImageNet,G(Ii,j) denotes the image Glass-Net produces and C denotes the ground truth. And the adversarial loss for the Glass-Net generator can be expressed as:

$$L_{adv} = -\frac{1}{HW} \sum_{i=1}^{H} \sum_{j=1}^{W} D(G(I)_{i,j})$$

where D(·) denotes the output of the discriminator. the four loss functions are incorporated into a hybrid loss Lhybrid for finetuning our generator Glass-Net with appropriate weights. The hybrid loss is defined as:

$$L_{hybrid} = 0.5 \times L_1 + 0.05 \times L_{SSIM} + 0.1 \times L_{VGG} + L_{adv}$$

Reimplemented Instance Normalization:
operators supported by tflite framework are used to reimplement IN to make sure all model ops compute on smartphone GPU.
results:

Fig. 3. From left to right: input images, results of Dutta et al., results of PyNET, our results. Some pictures show that the method of Dutta et al. and PyNET fail to separate front scenes from back scenes due to bad depth maps obtained from Megadepth.

team/method	PSNR	SSIM	MOS
Dutta et al.[3]	22.14	0.8778	-
xuehuapiaopiao-team	22.97	0.8758	4.1
Terminator	23.04	0.8756	4.1
CET_CVLab	23.05	0.8591	3.2
Team Horizon	23.27	0.8818	3.2
PyNET[8]	23.28	0.8780	4.1
Zheng et al.[9]	23.44	0.8874	-
AIA-Smart	23.55	0.8829	3.8
IPCV_IITM	23.77	0.8866	2.5
Ours	23.58	0.8770	**4.2**

Table 1. The results on the EBB! test subset obtained with different solutions.

Paper 3: " Efficient Multi-Lens Bokeh Effect Rendering and Transformation "

In this paper the authors aim to improve mobile photography, therefore it is also important to address the method complexity considering the computational limitations of mobile devices. Ignatov et al. studied efficient Bokeh rendering for mobile devices, being able to deploy the models on different target platforms. These challenges use the popular large-scale Everything is Better with Bokeh! (EBB!) dataset containing more than 10 thousand images collected in the wild. By controlling the aperture size of the lens, pairs of images with wide (aperture f/16) and shallow (aperture f/1.8) depth-of-field were taken, resulting in a normal sharp photo and one exhibiting a strong Bokeh effect. In this work, they propose a novel neural rendering method able to control the effect by feeding the lens properties into the neural network as an additional input.

They design their network EBokehNet for Bokeh effect rendering and transformation considering the following desired features: (i) the network should allow control of the strength and style of the Bokeh effect. This is fundamental to tackle the novel Bokeh Effect Transformation task. (ii) The model must be efficient in order to be usable. To achieve this, they adopt the already efficient NAFNet architecture and simplify it further by reducing the number of encoder-decoder blocks. (iii) The model should be able to render or convert the Bokeh effect of one lens to the effect of another lens without modifying the sharp foreground regions in the image. This ultimately implies a SOTA performance.

They train all the models using Adam optimizer with Cosine Annealing learning rate scheduler with 10 epochs of linear warm-up using a maximum learning rate of 1e-3 and minimum learning rate of 5e-5. They train the models to convergence, for the small model 220 epochs, and 280 for the large version. They use simple L1 loss. Since the small model is quite efficient, they can use high-resolution crops of 1024x1024 on images for training, meanwhile they use 512×512 crops for the larger version. They apply standard augmentations consisting of flips and rotations. They set the mini-batch size to 14 and run a distributed training over 7 RTX 3090Ti GPUs via DDP. The complete training time is approximately 48hrs per model. Furthermore, to increase the inference performance on the full-resolution images, they employ a Test Time Local Converter [6]. Fine-tuning on EBB. They use the aforementioned experimental setup with the following modifications. They start with a learning rate of 5e-4. They keep the same architecture and disable the lens and Bokeh strength encoding; therefore, they use only baseline blocks. They just need to train 50 epochs to achieve SOTA results.

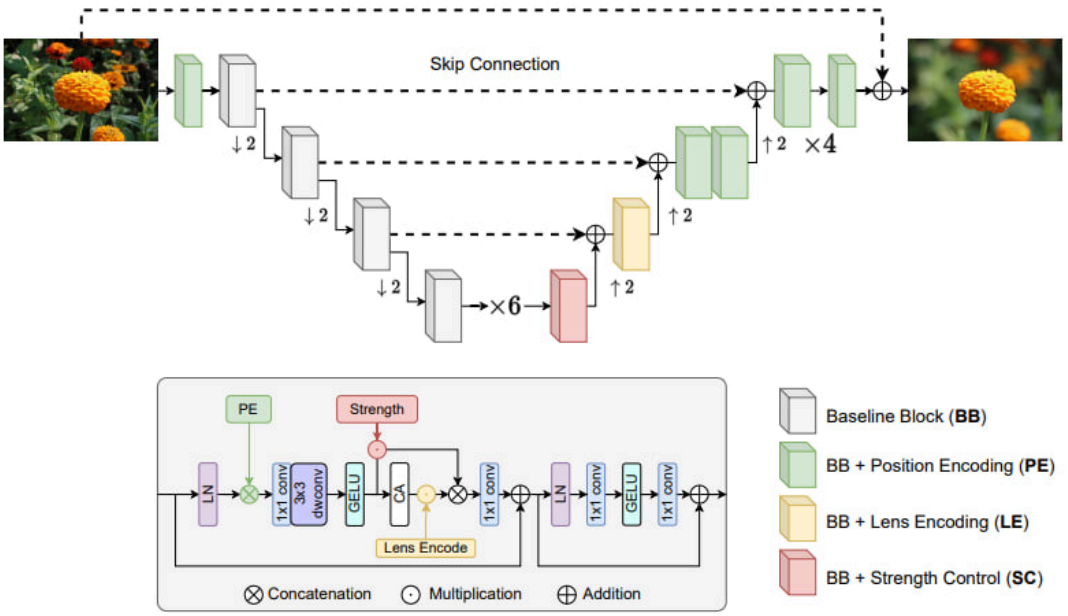
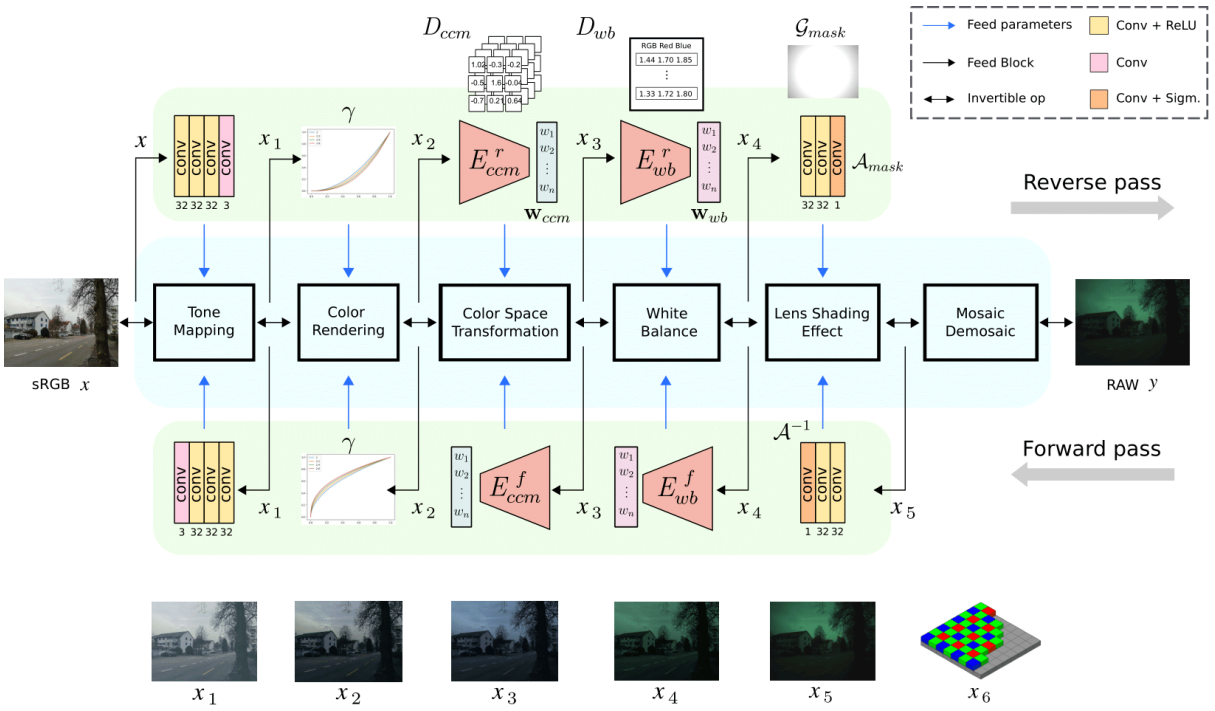


Figure 2. Architecture of the proposed EBokehNet. We use a encoder-decoder structure inspired in NAFNet [4]. We propose a new modified `Baseline Block`. We inject the lens information (type and strength) at different stages -as shown in the colour legend-, by doing this we are able to control the shape and strength of the Bokeh effect by "conditioning" the network's features. Additionally, we employ 2D positional encoding (PE) in some blocks [24] to provide extra spatial context. All the indicated operations are channel-wise.



Metrics compared to other models:

Method	PSNR ↑	SSIM ↑	LPIPS [45] ↓
EBokehNet	**24.99**	0.852	0.1912
SKN [22]	24.66	0.8521	0.3323
DBSI [9]	23.45	0.8675	0.2463
DMSHN [10]	24.72	0.8793	0.2271
DDDF [30]	24.14	0.8713	0.2482
BGGAN [31]	24.39	0.8645	0.2467
BRADCN [23]	24.83	0.8737	**0.1448**
PyNet [14]	24.93	0.8788	0.2219
BEViT [38]	24.57	0.8880	0.1985

Table 2. Quantitative results on the **EBB** [14, 16] **Val294** testset. Some numbers are borrowed from [23, 38].

Paper 4: "Using Depth Mapping to realize Bokeh effect with a single camera Android device"

This paper seeks to produce a bokeh effect with a single image taken from an Android device by post processing. Depth mapping is the core of Bokeh effect production. They present algorithms to determine the defocus map from a single input image. They obtain a sparse defocus map by calculating the ratio of gradients from original image and reblured image. Then, a full defocus map is obtained by propagating values from edges to the entire image by using nearest neighbor method and matting Laplacian. Based on the defocus map, foreground and background segmentation is carried out. They show that by enhancing the blurriness of the background while preserving the foreground, bokeh effect is achieved. Finally, they propose an algorithm to add a cartoon-like effect on the input image.
Bokeh is the shallow-depth of field effect which blurs the background of portrait photos (typically) to bring emphasis to the subject in the foreground.

Defocus Blur concept

When a picture is taken, portions of the scene that lie on the focal plane will look sharp while portions of the image that are away from the focal plan will appear blurry. The blurriness is associated with defocus when the scene is imaged on camera sensor. Light rays from points in the scene that are in-focus will reach points on the sensor whereas light rays from points that are out-of-focus will be spread across multiple points on the sensor within an area known as circle of confusion. The size (diameter) of the circle of confusion is associated with the amount of defocus for a given point in the scene and is given by:

$$c = \frac{|d - d_f|}{d} \frac{f_0^2}{N(d_f - f_0)}$$

where f d is distance between camera lens and focal plane, d is distance between a point in scene to the camera lens, 0 f is focal length, and N is stop number of the camera.
This leads to the out-of-focus points being imaged with a blur effect known as defocus blur. The defocus blur can be approximated as a Gaussian blur (with standard-deviation σdb) which varies with the distance of the object (d). They use a method, in which a given image is re-blurred using a Gaussian kernel with standard deviation σr . Gradient magnitude is computed for both the original image and the re-blurred image. The ratio of the gradient magnitude (R) of the original image to the re-blurred image is calculated. Sharper in-focus objects in the image will result in large R values and blurry out-of-focus objects will result in smaller R values, so R will reach maximum value at edge locations. Finally defocus-blur is estimated as follows:

$$R = \frac{|\nabla C|}{|\nabla C_r|} = \sqrt{\frac{\sigma_{db}^2 + \sigma_r^2}{\sigma_{db}^2}}$$

$$\sigma_{db} = \frac{1}{|\sqrt{R^2 - 1}}$$

They first convert the image to gray-scale to obtain intensity map. Then they compute gradients in x- and y-directions and take the magnitude at every pixel. They do the same for the re-blurred image and take gradient-ratio at every pixel. They also apply a mild median filter to reduce noise in the gradient magnitude

$$||\nabla C(x,y)|| = \sqrt{\nabla C_x^2 + \nabla C_y^2}$$

They perform canny edge-detection on the original image to a binary edge map. They modify the threshold generated by Matlab to take the lower threshold value equal to 85% of the upper threshold value.

They use the binary edge map to obtain a sparse gradient ratio map. They convert this to sparse Defocus map using eq. (3). This sparse defocus map can still contain inaccuracies due to noise and weak edges in the image. To mitigate this they apply a bilateral filter to the sparse defocus map. The bilateral filter reduces noise using a spatial Gaussian kernel without heavily blurring the edges.

<u>Propagation to full defocus map with Matting-Laplacian (elaborated in more details in Paper-5)</u>

They also evaluated an approach for interpolation that uses the matting-Laplacian matrix (L) to obtain full defocus map [6]. A matte (α) for a given image is a transparency mask that ranges between 0 and 1 which can be used to separate background from foreground.

$$C_i = \alpha_i F_i + (1 - \alpha_i) B_i,$$

where Ci is the RGB image (color components at pixel i), Fi is foreground, Bi is background, and αi is the matte.

So far we have discussed obtaining the defocus map. The defocus blur is directly related to the size of the circle of confusion (σ_db = k * c). Combining with eq.

$$c = \frac{|d - d_f|}{d} \frac{f_0^2}{N(d_f - f_0)}$$

one can obtain the depth map (distance map) of each pixel in the image provided the camera parameter values are known.

After depth map is available, images can be segmented into different segmentations. To achieve Bokeh effects, only 2 segments are applied here, resulting in a foreground and background. The original RBG image is blurred using a Gaussian kernel. The binary mask of the background will be applied on the blurred RBG image and the binary mask of the foreground will be applied on the original RBG image. Finally, the two images are added together to produce the image with Bokeh effect.

Paper 5: "A Closed Form Solution to Natural Image Matting"

This paper is authored by Anat Levin, Dani Lischinski, and Yair Weiss from the School of Computer Science and Engineering, Hebrew University. Interactive digital matting is the process of extracting a foreground object from an image based on limited user input. To do so we must estimate the foreground and the background colors, as well as the foreground opacity ("alpha matte") from a single color measurement.

The color of the i-th pixel is assumed to be a linear combination of the corresponding foreground and background colors,

$$I_i = \alpha_i F_i + (1 - \alpha_i) B_i,$$

where α_i is the pixel's foreground opacity. In natural image matting, all quantities on the right hand side of the compositing equation(the equation above) are unknown.

(a) (b) (c) (d) (e) (f)

In above figure, we have: (a) An image with sparse constraints: white scribbles indicate foreground, black scribbles indicate background. Applying Bayesian matting to such sparse input produces a completely erroneous matte (b). Foreground extraction algorithms, such as produce a hard

segmentation (c). An automatically generated trimap from a hard segmentation may miss fine features (d). An accurate hand-drawn trimap (e) is required in this case to produce a reasonable matte (f).

To derive our solution for the grayscale case we make the assumption that both F and B are approximately constant over a small window around each pixel. In that case we have,

$$\alpha_i \approx aI_i + b, \quad \forall i \in w,$$

Our goal in this paper will be to find α, a and b minimizing the cost function,

$$J(\alpha, a, b) = \sum_{j \in I} \left(\sum_{i \in w_j} (\alpha_i - a_j I_i - b_j)^2 + \varepsilon a_j^2 \right),$$

where wj is a small window around pixel j.

A simple way to apply the cost function to color images is to apply the gray level cost to each channel separately. Alternatively we can replace the linear model we had for grayscale images, with a 4D linear model:

$$\alpha_i \approx \sum_c a^c I_i^c + b, \quad \forall i \in w$$

The advantage of this combined linear model is that it relaxes our previous assumption that F and B are constant over each window. Instead, as we show below, it is enough to assume that in a small window each of F and B is a linear mixture of two colors; in other words, the values Fi in a small window lie on a single line in the RGB color space: Fi = βiF1 + (1 − βi)F2, and the same is true for the background values Bi . In what follows we refer to this assumption as the color line model.

In this system the user-supplied constraints on the matte are provided via a scribble-based GUI. The user uses a background brush (black scribbles in our examples) to indicate background pixels (α = 0) and a foreground brush (white scribbles) to indicate foreground pixels (α = 1). To extract an alpha matte matching the user's scribbles we solve for

$$\alpha = \text{argmin } \alpha^T L \alpha, \quad \text{s.t. } \alpha_i = s_i, \ \forall i \in S$$

where S is the group of scribbled pixels and si is the value indicated by the scribble.

(a) (b) (c) (d)

In above figure matting examples are shown. (a,c) Input images with scribbles. (b,d) Extracted mattes.

The matting Laplacian matrix L is a symmetric positive definite matrix. This matrix may also be written as L = D−W, where D is a diagonal matrix D(i,i) = ∑jW(i, j) and W is a symmetric matrix, whose off-diagonal entries are defined by

$$\sum_{k|(i,j)\in w_k} \left(\delta_{ij} - \frac{1}{|w_k|} \left(1 + (I_i - \mu_k)(\Sigma_k + \frac{\varepsilon}{|w_k|} I_3)^{-1} (I_j - \mu_k) \right) \right)$$

Thus, the matrix L is the graph Laplacian used in spectral methods for segmentation, but with a novel affinity function given by the above equation.

In contrast, by rewriting the matting Laplacian as L = D−W we obtain the following affinity function, which this paper refers to as “the matting affinity”:

$$W_M(i,j) = \sum_{k|(i,j)\in w_k} \frac{1}{|w_k|} \left(1 + (I_i - \mu_k)(\Sigma_k + \frac{\varepsilon}{|w_k|} I_3)^{-1} (I_j - \mu_k) \right)$$

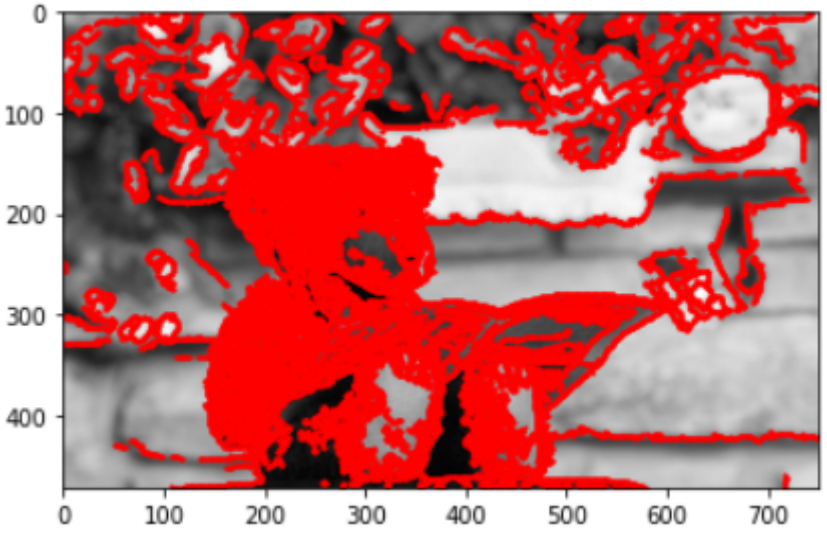
The alpha matte may be predicted by examining the smaller eigenvectors of the matting Laplacian, since an optimal solution to “**α = argmin α T L α**” will be to a large degree spanned by the smaller eigenvectors.

Implementation

As mentioned before, the method we used to implement this project is based on 'Paper 4' in the Literature Review section.

For input image:

Initially we applied a grayscale filter to reduce the number of image channels to one. Subsequently, we utilized a Canny edge detector to compute edge pixels and determined the ratio of the magnitude of the gradient in the original image to a blurred version of the image at the edge pixels. We then calculated the defocus blur at edge pixels, generating a sparse defocus map.



After obtaining the sparse defocus map, we employed the matting Laplacian method to interpolate values and obtain a full defocus map. Additionally, we experimented with another method for calculating the defocus map, available on GitHub, and found its results more suitable. Therefore, we adopted that implementation.

The next page features two images: the Defocus Blur Maps resulting from method 1 and method 2, presented in order.

Defocus Blur Map with first method

Defocus Blur Map with second method

Finally, we normalized the defocus map and identified pixels with defocus less than the median of defocus values as the foreground. The output image with the Bokeh effect was eventually created by applying a Gaussian filter to only the background and then attaching the foreground.

Bokeh Effect Applied

Results and Conclusion

The adopted implementation has a significant advantage, as evidenced in the image.

However, a challenge with the foreground mask is its lack of connectivity. Therefore, an optimization idea is to compute the connectivity components and select the largest component as the true foreground.

The issue with the foreground mask negatively impacts the overall quality of the Bokeh effect, as can be observed in the image below.

This method exhibits inefficiencies when applied to images with insignificant defocus blur or when the main object is not sufficiently closer to the camera than the background.

[Image URL]

In this particular case, there is no significant difference between the defocus value of the foreground and some background regions.

Consequently, the foreground map lacks the precision needed.