Self-Supervised Video Defocus Deblurring with Atlas Learning

Lingyan Ruan
lyruan@mpi-inf.mpg.de
Max-Planck-Institut für Informatik
Saarbrücken, Germany

Martin Bálint
mbalint@mpi-inf.mpg.de
Max-Planck-Institut für Informatik
Saarbrücken, Germany

Mojtaba Bemana
mbemana@mpi-inf.mpg.de
Max-Planck-Institut für Informatik
Saarbrücken, Germany

Krzysztof Wolski
kwolski@mpi-inf.mpg.de
Max-Planck-Institut für Informatik
Saarbrücken, Germany

Hans-Peter Seidel
hpseidel@mpi-inf.mpg.de
Max-Planck-Institut für Informatik
Saarbrücken, Germany

Karol Myszkowski
karol@mpi-inf.mpg.de
Max-Planck-Institut für Informatik
Saarbrücken, Germany

Bin Chen∗
binchen@mpi-inf.mpg.de
Max-Planck-Institut für Informatik
Saarbrücken, Germany
University of Melbourne
Melbourne, Australia

∗Corresponding author

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike International 4.0 License.

SIGGRAPH Conference Papers ’24, July 27–August 01, 2024, Denver, CO, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0525-0/24/07
https://doi.org/10.1145/3641519.3657524

ABSTRACT

Misfocus is ubiquitous for almost all video producers, degrading video quality and often causing expensive delays and reshoots. Current autofocus (AF) systems are vulnerable to sudden disturbances such as subject movement or lighting changes commonly present in real-world and on-set conditions. Single image defocus deblurring methods are temporally unstable when applied to videos and cannot recover details obscured by temporally varying defocus blur. In this paper, we present an end-to-end solution that allows users to correct misfocus during post-processing. Our method generates...
and parameterizes defocused videos into sharp layered neural atlases and propagates consistent focus tracking back to the video frames. We introduce a novel differentiable disk blur layer for more accurate point spread function (PSF) simulation, coupled with a circle of confusion (COC) map estimation module with knowledge transferred from the current single image defocus deblurring (SIDD) networks. Our pipeline offers consistent, sharp video reconstruction and effective subject-focus correction and tracking directly on the generated atlases. Furthermore, by adopting our approach, we achieve comparable results to the state-of-the-art optical flow estimation approach from defocus videos.

CCS CONCEPTS
- Computing methodologies → Computational photography.

KEYWORDS
defocus deblur, implicit representation, neural atlas, video deblur, post refocus

ACM Reference Format:

1 INTRODUCTION

Shallow focus initially emerged due to early filmmaking constraints such as limited ISO sensitivity. However, directors soon leveraged its storytelling potential to direct the audience’s attention. Thus, shallow focus quickly became essential for achieving a cinematic look. Today, professional focus pullers meticulously plan for shoots, particularly long takes or “oners”, involving extensive camera movement and complex blocking. Maintaining focus is crucial for a successful shot.

Conversely, amateur content creators rely on camera autofocus (AF). Modern AF systems have advanced from traditional Contrast Detection (CDAF) [Chen and van Beek 2015; Vuong and Lee 2013] and Phase Detection (PDAF) [Fontaine 2017; Inoue and Takahashi 2009] to smarter systems with face, eye, and object tracking capabilities [Wang et al. 2021]. Unfortunately, in real-world scenarios, abrupt changes in subject movement, lighting conditions, or the presence of multiple subjects can easily disrupt AF systems, resulting in misfocused footage.

Technical deficiencies and human mistakes frequently force video producers to either reshoot their misfocused footage or settle for lower-quality videos, ignoring the focus errors. Restoration of misfocused footage in post-production is emerging as a promising alternative. Our paper introduces a pipeline designed to reconstruct sharp content from videos distorted by defocus, enable focus editing to fix subject tracking, allow for convenient scene editing via our learned atlas, and output additional channels that enable complex effects during compositing.

In Fig. 2, we illustrate the complexity of our task by simulating a video featuring three spatial points: a stationary green point in the foreground, a stationary blue point in the background, and a dynamic orange point moving freely in 3D space. Each stationary point has a unique defocus level determined by its depth in the scene. As the focus distance shifts from front to back (light blue rectangles) over time, the defocus degree for each stationary point also changes. In real-world video capture scenarios, camera movement adds another layer of ambiguity. These complexities pose significant challenges to video defocus deblurring and influence other video processing tasks such as object tracking or optical flow estimation [Ruan et al. 2021; Teed and Deng 2020].

Correcting misfocus is a two-step procedure that includes eliminating defocus blur, also known as defocus deblurring, followed by defocus synthesis, which applies defocus blur following the scene’s depth. Defocus synthesis is a forward problem and has been investigated for images [Srinivasan et al. 2018; Wadhwa et al. 2018] and videos [Zhang et al. 2019] in the past. Defocus deblurring, on the other hand, is an inverse problem and poses different challenges than defocus synthesis. Misfocus correction in video, unlike single image defocus deblurring (SIDD), requires an emphasis on maintaining temporal consistency. The key difficulty here is the temporally varying defocus blur caused by changes due to scene and camera motion and, potentially, autofocus camera functionality.

Inspired by recent implicit video representation approaches [Kasten et al. 2021], we apply the layered atlas to parameterize and deblur our video with good consistency and temporal stability. Recently, Abuolaim et al. [2021] synthesized a dataset based on the sharp images and depth maps in the SYNTHIA dataset [Hernandez-Juarez et al. 2019]. However, when applied to real-world videos, methods trained on synthetic datasets often suffer due to the domain gap. Our self-supervised network circumvents the need for such extensive datasets and performs deblurring directly on each individual video.

In this work, we make the following contributions: (1) We introduce the first end-to-end video defocus deblurring approach, allowing for extensive scene editing, including focus corrections and bokeh editing. (2) We implement a lens blur CUDA layer featuring a novel differentiable disk kernel capable of simulating the realistic fall-off boundary (soap bubble effect) of the Point Spread Function (PSF). (3) We introduce a COC map estimation network using a transfer learning approach. This strategy facilitates a more lightweight architecture while maintaining competitive performance.
(4) As a direct application of our pipeline, we show that focus correction and tracking can be conducted on our output foreground and background channels, our learned atlases allow for extensive editing, and we demonstrate that optical flow can also be estimated from our learned UV map.

Our code and the training data we rendered with synthetic defocus are available under https://neuralatlasvdd.mpi-inf.mpg.de.

2 RELATED WORK

In this section, we cover two main tracks of work that are most related to our approach: the image/video defocus deblurring and implicit image/video representations.

2.1 Image/Video Defocus Deblurring

Dealing with defocus blur has been a persistent challenge in image processing and computer vision, largely because of its inherent spatially varying nature in static and dynamic scenes compared to motion blur. Recovering the sharp latent details and information from blurred images or videos holds significant potential for various applications, including object detection [Redmon and Farhadi 2018] and text recognition [Liao et al. 2021]. The majority of prior research has concentrated on SIDD.

**Single Image Defocus Deblurring.** Traditional solutions typically follow a two-step process involving estimating a dense defocus map followed by non-blind deconvolution [D’Andrés et al. 2016; Karaali and Jung 2017; Park et al. 2017; Shi et al. 2015]. In this process, the quality and precision of the defocus map are critical factors significantly impacting the final outcome. Techniques utilizing deep neural networks to estimate defocus maps have been proposed [Karaali et al. 2022; Lee et al. 2019]. Recent works such as [Zhang and Sun 2021] and [Piché-Meunier et al. 2023] have advanced the estimation process by jointly deriving depth and defocus while adhering to consistency constraint, where deriving the lens parameters leads to further improvements [Piché-Meunier et al. 2023].

Our work aligns with their goals in accurately determining the circle of confusion (COC) [Potmesil and Chakravarty 1982] for realistic defocus synthesis. However, we diverge from these methods by omitting scene physical depth estimation and estimating defocus level in pixel units, which better suits our pipeline.

The advent of deep learning has significantly advanced SIDD, leading to the emergence of neural network-based solutions. Abouab et al. [2020] introduce a dual-pixel defocus deblurring dataset and an end-to-end network to recover a sharp image from its defocused counterpart. Ruan et al. [2021] introduce another dataset synthesized from light field images to address the defocus image and sharp ground truth mismatch problem. Other approaches have integrated spatially varying blur into network structures with per-pixel dynamic residual kernels [Ruan et al. 2022], iterative filter adaptive kernels [Lee et al. 2021], learnable recursive kernels [Quan et al. 2023], Gaussian kernel mixture [Quan et al. 2021] in an end-to-end manner, more recent studies, e.g., [Zamir et al. 2022] have incorporated transformer structures. Unfortunately, most of these methods present as a black box providing no straightforward ways to obtain COC maps, use temporal data available in other frames of the videos, or map to a temporally consistent space such as our multi-layer atlas.

**Video Defocus deblurring.** Unlike single image defocus deblurring or synthesis, blurry videos present additional challenges, particularly due to temporal variations discussed in Sec. 1. This area remains less explored but also holds significant practical appeal. RDPPD [Aboulaim et al. 2021] proposes an RNN-based network structure to handle defocus deblurring for image sequences. Neucam [Huang et al. 2023] proposes an implicit camera model to simulate the image signal processing (ISP) process in a deep neural network and can recover all-in-focus images from multi-focus stacks. We compare our approach with theirs. Although no video defocus deblurring is demonstrated in the original Neucam paper, we adapt their video motion deblurring network to video defocus deblurring for a fair comparison. It is worth noting that video defocus deblurring is different from the problem of sharp image reconstruction from multi-focus, which is based on the assumption that the camera and scene are static. Several methods address videos involving dynamic scenes with static cameras using custom-designed camera systems [Zhou et al. 2012], specialized optical systems like deformable lenses [Miau et al. 2013], or rely on known focus distances [Kim et al. 2016]. We relax these constraints in our approach.

2.2 Implicit image/video representation

Emerging works integrate scene representation as neural radiance fields (NeRF) with multiple layer perceptrons (MLPs) in a continuous manner for 3D geometry [Mildenhall et al. 2021]. NeRF reconstruction quality degrades when input images contain defocus blur (shallow depth of field) [Wu et al. 2022]. To address this, Wu et al. [2022] incorporate explicit aperture modeling, coupled with a differentiable COC representation, while Li et al. [2022] instead incorporate a learned deformable kernel into the pipeline to compensate for defocus blur. A recent work [Wang et al. 2023] investigates reconstructing the all-in-focus frame from the image stack by simultaneously estimating depth constrained by an explicit physical camera model. Unfortunately, all these methods are limited to static scenes.

Motivated by recent success in dynamic video representation, such as view synthesis [Li et al. 2023], volumetric video rendering [Peng et al. 2023], video decomposition [Kasten et al. 2021], integrating MLPs to overfit to video content, we aim to fit a unique sharp content that could represent the whole sequence, thus achieving the spatially and temporally consistency for misfocused video. Specifically, we adopt a 2D atlas representation, which parameterizes the dynamic and static background separately with MLPs, originally used for video consistent editing, allowing alterations on the atlas to be reflected in the original video. This enables several applications such as text-driven video stylization [Bar-Tal et al. 2022], sketch face editing [Liu et al. 2022], video de-flickering [Lei et al. 2023], and others.

3 METHOD

Aiming for consistent video editing, layered neural atlases decompose an input video into a collection of 2D layered atlases [Kasten et al. 2021]. This representation effectively circumvents the need to
physically resolve the ambiguity between camera and object movement in our problem (Fig. 2) by efficiently mapping the motion of dynamic points onto a 2D plane. The multi-layer representation can be seamlessly adapted to our specific challenge of distinguishing between static and dynamic points, allowing for separate treatment.

Also, the layered neural atlases enable straightforward and intuitive editing using readily available image editing tools, which can be naturally propagated back to the video frames consistently. The distinct foreground, background, and mask layers enable us to conduct focus-tracking in post-production and are easy and quick to compose into video frames.

The efficient representation of video, intuitive editing space, and consistent editing quality make layered neural atlases excel in sharp video editing. However, it does not demonstrate strong performance when applied directly to video defocus deblurring tasks. We demonstrate and compare the performance of the original layered neural atlas and the combination of it and our reblurring network results in inferior performance. Inspired by Deep Image Prior (DIP) [Lempitsky et al. 2018] and [Ye et al. 2022], we employ a CNN network for our Atlas Learning Network.

3.2 Atlas Learning

Deep Image Prior (DIP) uses the structure of a deep convolutional network to implicitly capture prior knowledge about natural image statistics and has proven to be particularly suitable for image restoration tasks [Lempitsky et al. 2018]. In general, a simple MLP structure tends to perform less effectively in capturing spatial features such as textures and edges when compared to CNN structures without specialized design [Tolstikhin et al. 2021; Tu et al. 2022]. Thus, we employ a lightweight CNN to generate and represent our layered atlas images from noise maps. We have observed that a decoder-only architecture efficiently learns our atlases. We applied pixel-shuffle [Shi et al. 2016] in the upsampling layers to better learn and preserve high-frequency details. The results indicate improved performance compared to the original MLP-based atlas representation, as elaborated in Sec. 4.2.

The Atlas Learning Network, denoted as \( A \), takes as input the noise map \( \mathbf{Z}_f, \mathbf{Z}_b \) ∈ \( \mathbb{R}^{C' \times H' \times W' \times T} \) and upscals it to produce our foreground and background atlas images \( \mathbf{A}_f, \mathbf{A}_b \) ∈ \( \mathbb{R}^{3 \times H \times W \times T} \). Note that the Atlas Learning Network shares weights between foreground and background atlas generation. Given a particular point

\begin{equation}
\mathbf{Z}_f, \mathbf{Z}_b \rightarrow \mathbf{A}_f, \mathbf{A}_b
\end{equation}
p, we employ the acquired $UV_f^p$ and $UV_b^p$ coordinates to sample colors from $A_f$ and $A_b$. Finally, we obtain the latent sharp color $c_{\text{sharp}}^p$ by combining these two colors, using the learned $\alpha$ value as the blending factor. This entire procedure can be summarized as follows:

$$c_{\text{sharp}}^p = \alpha A_f(\psi_f(p)) + (1 - \alpha) A_b(\psi_b(p))$$

where $A_f = A(Z_f)$ and $A_b = A(Z_b)$. Note that we utilize one foreground and one background layer to present the video content in this paper. However, additional layers can readily be implemented to accommodate more complex motion and scenes as indicated in [Kasten et al. 2021]. To encourage the network generating sharp images, we incorporate a reblur module to reproduce the blurred images, allowing the network to be optimized using the original input video frames as self-supervision.

### 3.3 Lens Blur Layer

To enable reblurring of the latent sharp image, we have developed a differentiable disk kernel CUDA layer that supports the continuous radius of PSF in our reblur module. While the disk kernel is considered a more accurate representation of realistic lens blur compared to the Gaussian kernel [Potmesil and Chakravarty 1982], it presents a challenge in terms of differentiability. Traditional disk kernels are typically represented in discrete sizes, making them unsuitable for accurately representing a continuous PSF radius. Existing methods for achieving differentiability with the disk kernel typically rely on functions like the hyperbolic tangent (tanh) as demonstrated in [Busam et al. 2019] and [Luo et al. 2023], or piecewise functions as discussed in [Gwosdek et al. 2011]. These approaches primarily focus on controlling the size of the PSF and result in uniform PSFs. However, the actual shape of the PSF is non-uniform, with sharply defined fall-off edges due to optical aberrations, as highlighted in [Tang and Kutulakos 2012]. Abuolaim et al. [2021] attempted to create the fall-off boundary of the PSF by combining a Butterworth filter with an undifferentiable disk kernel, resulting in a final kernel that remains undifferentiable. In this paper, we propose a straightforward and efficient solution—a differentiable disk kernel that inherently captures the sharply defined fall-off boundary without the need for additional modifications or complex functions. The value $k$ inside our differentiable disk kernel can be calculated as:

$$k = \frac{e^{\beta d} - e^d}{e^{\beta d} - 1} \quad \text{with} \quad d = r - \sqrt{x^2 + y^2},$$

where $d \in \mathbb{R}$ represents the distance to the edge of disk, $r \in \mathbb{R}^+$ the radius of disk, and $x, y \in [-\lfloor \frac{s}{2} \rfloor, \lfloor \frac{s}{2} \rfloor] + 1$ with $s \in \mathbb{N}$ representing the kernel size. We use $\beta \in \mathbb{R}^+$ to control the thickness of the fall-off boundary. Note that the value of $\beta$ should be greater than $\frac{1}{2}$ to ensure a positive kernel. We used $\beta = 2$ in our lens blur CUDA layer. We applied per-pixel scatter-sum convolution similar to [Our and Wolf 2019]. With our lens blur layer, the sharp image can be reblurred as:

$$\hat{c}_{\text{blur}}^p = L(c_{\text{sharp}}^p, r^p)$$

For simplicity and efficiency, we construct the estimated sharp image using two layers. Optionally, multiple layers can be used, composing from back to front, to better handle occluded areas around the subject’s silhouette, as demonstrated in [Zhang et al., 2019]. As depicted in Fig. 4, we conduct a comparative analysis involving our differentiable disk kernel, the Gaussian kernel, an undifferentiable disk kernel, and methods from [Gwosdek et al. 2011] and [Busam et al. 2019]. We show real sharp and blurred images (GT sharp and GT blur) and their reblurred versions using these kernels. In the third row, we vary our differentiable disk kernel’s radius $r$, and in the last row, we demonstrate the fall-off effect controlled by $\beta$.

### 3.4 COC Map Estimation

Our lens-blur CUDA layer requires per-pixel input regarding the radius of COC. Typically, a physical thin lens model is employed to compute COC from depth, which relies on camera-specific parameters when capturing this particular video, such as aperture size, focal length, and focus distance. Capturing depth and such metadata would greatly limit the applicability of our method. In our pipeline, we address this challenge by training a network $\mathcal{C}$ to directly estimate the COC radius $r^p = \mathcal{C}(c_{\text{blur}}^p)$ in pixel units from each frame. To this end, we employ the concept of transfer learning and draw upon encoded defocus knowledge from existing defocus deblurring networks. Specifically, we train a defocus map estimation network structured similarly to [Ruan et al. 2023] due to its lightweight and efficient design, which aligns with our requirements. We fix the weights of their encoder, which encodes...
rich defocus information after being trained on large datasets, and exclusively train the modified decoder to estimate the defocus map rather than a sharp image. We show the effectiveness of this strategy in Sec 4.2 and generalize well on real image defocus estimation. To supervise the network training, we create a large defocus map estimation dataset, which includes rendered defocus images and corresponding ground truth COC maps. We select 22 animations from Blender Open Movies [Blender 2024] and render 25 distinct defocus variations for each frame, encompassing 5 different defocus levels (aperture sizes) and 5 focus distances. In total, we prepare a dataset comprising 27K pairs of defocus and COC maps, where COC map is obtained based on the focal distance, f-number and depth map.

3.5 Loss

In training our network, we incorporate a subset of loss functions from the original layered neural atlas approach. These include \( L_{\text{rigid}} \), which ensures a rigid mapping to the atlas for intuitive editing through the Jacobian matrix; \( L_{\text{flow}} \), which minimizes disparities between corresponding points in the video, ensuring consistency; and \( L_{\text{sparsity}} \), which prevents redundant representations in both foreground and background atlases. Please refer to [Kasten et al. 2021] for more details. We combine our losses as follows:

\[
\mathcal{L} = \sum_{i=1}^{n} (\lambda_1 L_{\text{reco}} + \lambda_2 L_{\text{flow}} + \lambda_3 L_{\text{sparsity}} + \lambda_4 L_{\text{rigid}}), \quad (4)
\]

Specifically, for \( L_{\text{reco}} \), between the blurry input video and re-blurred frames as shown in Fig. 3, we use the \( L_1 \) loss, together with multi-scale structural similarity index (MS-SSIM) as suggested in [Zhao et al. 2016] for the image restoration. \( \lambda_n \) are our weights of losses during training, the choice of which we explain in the following section.

4 EXPERIMENTS

4.1 Implementation details

We evaluate our pipeline using videos with dimensions of 512 × 288, each consisting of approximately 60 frames. The training process was conducted on an NVIDIA Quadro RTX 8000, employing the Adam optimizer with a learning rate of 3e-4 over 50,000 iterations. To control the loss functions, we set the weights \( \lambda_1 \) to \( \lambda_4 \) to 0.3, 1.5, 100, and 0.5, respectively. We set the atlas resolution at 640 × 360, with the input noise dimension being eight times smaller than that of the atlas.

In line with the original layered neural atlas approach, we performed preliminary estimations of optical flow using RAFT [Teed and Deng 2020] and generated masks using SAM [Kirillov et al. 2023] for the first frame and XMem [Cheng and Schwing 2022] for subsequent frames. The training duration ranged from 6 to 8 hours. When training the COC Map Estimation Network, we start with a learning rate of 3e-4 for 300,000 iterations, followed by a linear decay to 1e-6 over another 300,000 iterations.

To assess our pipeline’s effectiveness, we evaluated two types of videos: rendered and real. For rendered videos, we modified four clips sourced from Blender Open Movies [Blender 2024] (distinct from our COC training dataset) to simulate camera misfocus. We utilized five clips of real videos, two of which were captured using a Canon 6D II camera, while the remaining three were obtained from online sources.

4.2 Results & Comparisons

We compare our approach to RDPD (RSDP+ variant) [Abuolaim et al. 2021] - the first framework that tackles defocus deblurring on image sequences, and Neucam (video deblur variant) [Huang et al. 2023] - video deblurring that uses implicit camera model. Note that while Neucam does not test video defocus deblurring in their paper, in our experiments, Neucam always converges and learns defocus deblurring.

We present in-depth qualitative comparisons for real captured video and rendered animation in Fig. 9. Among the methods evaluated, RDPD performs the least effectively in defocus deblurring, as the defocus blur remains unchanged in the original video frames, as evidenced in the insets of each frame and the time-dimensional data slice. Neucam, while capable of recovering sharp information, introduces noticeable artifacts. In contrast, our approach excels in both the quality of sharp content reconstruction and consistency. It’s worth noting that small details, such as facial expressions, pose a challenge for all algorithms, including ours, as illustrated by the face of the truck driver in Fig. 9.

![Figure 5: Qualitative comparison of COC map estimation on our semi-synthetic images and BLB dataset [Peng et al. 2022].](image)

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR↑</th>
<th>SSIM↑</th>
<th>tPSNR↑</th>
<th>Flip ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neucam</td>
<td>25.17</td>
<td>0.86</td>
<td>0.13</td>
<td>28.64</td>
</tr>
<tr>
<td>RDPD</td>
<td>27.93</td>
<td>0.92</td>
<td>6.91</td>
<td>0.10</td>
</tr>
<tr>
<td>Ours</td>
<td>29.23</td>
<td>0.95</td>
<td>1.28</td>
<td>0.07</td>
</tr>
</tbody>
</table>

We add corresponding quantitative comparison only on rendered animation in Tab. 1, as the ground truth for real videos is not available. Besides the PSNR and SSIM, we also use perception-informed FoVVideoVDP [Mantiuk et al. 2021], Flip [Andersson et al. 2020], and tPSNR [Banitalebi-Dehkordi et al. 2016] for capturing the distortions over large areas. Our approach achieves the best performance across all metrics.

COC map estimation. We assess our COC map estimation network in comparison to the top two approaches, namely, DMENet [Lee et al. 2019] and Inversedof [Piché-Meunier et al. 2023] (IDOF), as depicted in Fig. 5 and summarized in Table 2. For quantitative
comparison, we perform the evaluation on two datasets: the BLB dataset [Peng et al. 2022], employing the Blender’s Cycles renderer, allowing for a realistic representation of defocus effects while providing absolute scale COC values. Additionally, we construct a semi-synthetic dataset (SSD) using Bokehme [Peng et al. 2022], combining the traditional physics-based rendering approach with a neural rendering method, as recommended by [Piché-Meunier et al. 2023], and apply it to real RGB images. We randomly select 300 images from the FiveK Dataset [Bychkovsky et al. 2011] and generate 15 unique defocus variations for each image. These variations include 5 different defocus levels and 3 distinct focus distances, resulting in a test dataset of 4.5k images. Note that our neural network was not trained on these datasets; they were employed exclusively for testing purposes.

Table 2 illustrates our method’s performance, which is comparable to IDOF, albeit with marginally lower numbers. IDOF utilizes three Transformer-based structures to estimate disparity, defocus, and per-pixel weights between them, enhancing accuracy at a cost approximately five times greater than our approach. Additionally, IDOF trained on a dataset of roughly 1120k images, whereas our strategy was trained on a more modest dataset of 37k images. This dataset comprised approximately 11k real images originally intended for single-image defocus deblurring and around 26k images rendered for training purposes.

### Table 3: Ablation study on the COC network training strategy evaluated on the two datasets using RMSE.

<table>
<thead>
<tr>
<th>Method</th>
<th>BLB PSNR</th>
<th>SSDF SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed encoder</td>
<td>8.77</td>
<td>2.80</td>
</tr>
<tr>
<td>Train from scratch</td>
<td>12.26</td>
<td>34.3</td>
</tr>
</tbody>
</table>

Our method, illustrated in Fig. 5, roughly matches the visual quality of IDOF and also adeptly manages difficult areas, such as glass. This performance mainly stems from the proficiency of our pre-trained encoder in extracting unique features from defocused images. This capability enhances the accuracy of defocus map estimation and shows strong generalization in real image defocus map estimation tasks. The ablation study results, detailed in Table 3, further support the effectiveness of this approach. Notably, the performance diminished when we trained the encoder from scratch using our rendered dataset rather than utilizing a pre-trained encoder.

### Table 4: Ablation study on the impact of the COC map estimation network in our pipeline. We present the quantitative assessment of reblurred and deblurred images of the truck scene.

```latex
\begin{tabular}{lcccc}
\hline
Metric & Baseline & DMENet & IDOF & Ours  \\
\hline
PSNR & 35.20/29.55 & 38.18/31.18 & 39.14/32.07 & 39.76/32.19  \\
SSIM & 0.917/0.796 & 0.962/0.872 & 0.977/0.915 & 0.978/0.914  \\
\hline
\end{tabular}
```

**Effect of COC map estimation network**: In addition to demonstrating the performance of the COC map itself with state-of-the-art (SOTA) algorithms, we also evaluate its impact on our pipeline. We compare it against alternative methods by either removing it entirely, identified as the baseline, or replacing it with existing COC estimation methods. We present results in terms of PSNR and SSIM for both reconstructed all-in-focus and reblurred images in Tab. 4.

The inaccurate COC map from DMENet leads to inferior performance, whereas our results are on par with IDOF regarding estimated COC (see Tab. 2) and translate to comparable restoration quality. This suggests that an accurate COC map is crucial for precise deblurring and reblurring performance.
5 APPLICATIONS

In this section, we showcase two practical applications that directly leverage our pipeline’s capabilities. Firstly, we illustrate how focus tracking for selected scene components can be seamlessly performed on the atlases generated by our system. Secondly, we demonstrate that our learned UV map enables the recovery of improved optical flow in defocused videos, achieving somewhat comparable performance of the state-of-the-art optical flow estimation algorithm described in [Teed and Deng 2020].

5.1 Focus Tracking

The layered neural atlas representation allows the refocusing process to be edited on a 2D atlas image, which is then mapped back to the original video frames. As demonstrated in Fig. 1, by keeping the foreground atlas and only adding blur to the background atlas, we can correct and simulate the focus tracking that failed during capturing. Please refer to the supplementary video.

5.2 Optical Flow Estimation

We approximate a linear transformation from an arbitrary atlas we can correct and simulate the focus tracking that failed during verse transform, computing flow. RAFT on neighbor frames and outperforms it on long-range optical frame intervals across four scenes. Our method is comparable to our method and RAFT [Teed and Deng 2020], testing on various coordinates results. We compute flow from frame further improves re-

\[ \begin{align*}
    \begin{bmatrix} x \\ y \\ t \end{bmatrix} &= J_{M}^{-1}(x', y', t) \begin{bmatrix} u \\ v \end{bmatrix} + o(x', y', t). 
\end{align*} \tag{5}
\]

Using Pytorch’s jacrev function, we compute and invert the Jacobian at a pixel \((x', y', t)\). We choose \((x', y', t)\) from all integer pixel coordinates in frame \(t\) such that \(M(x', y', t)\) is closest to \((u, v)\) by evaluating \(M\) on frame \(t\) followed by a nearest neighbour lookup. We compute the offset \(o\) as:

\[ o(x', y', t) = \begin{bmatrix} x' \\ y' \\ t \end{bmatrix} - J_{M}^{-1}(x', y', t)M(x', y', t). \tag{6} \]

Linearly interpolating the nearest inverse Jacobians and offsets with scipy.interpolate.LinearNDInterpolator further improves results. We compute flow from frame \(t_0\) to frame \(t_1\) by mapping pixel coordinates \((x, y, t_0)\) to \((u, v)\) using \(M\), and, with the described inverse transform, computing \((x, y, t_1)\) from the \((u, v)\) atlas points.

Table 5 and Fig. 7 compare the estimated optical flow between our method and RAFT [Teed and Deng 2020], testing on various frame intervals across four scenes. Our method is comparable to RAFT on neighbor frames and outperforms it on long-range optical flow.

**Table 5: Quantitative comparisons of optical flow: our method / RAFT [Teed and Deng 2020].**

<table>
<thead>
<tr>
<th>Interval</th>
<th>RMSE Error</th>
<th>EndPoint Error</th>
<th>Angular Error</th>
<th>Length Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.598/0.614</td>
<td>0.631/0.592</td>
<td>15.10/9.818</td>
<td>0.433/0.493</td>
</tr>
<tr>
<td>3</td>
<td>1.654/1.773</td>
<td>1.685/1.543</td>
<td>12.45/10.67</td>
<td>1.173/1.217</td>
</tr>
<tr>
<td>5</td>
<td>2.657/3.330</td>
<td>2.606/2.442</td>
<td>10.95/11.14</td>
<td>1.842/1.793</td>
</tr>
</tbody>
</table>

6 CONCLUSION

In conclusion, we propose a systematic end-to-end solution to video defocus deblurring and editing. By generating and parameterizing defocused video into layered neural atlases through a differentiable thin lens model within a self-supervised network, our method enables achieves consistent, sharp video reconstruction and focus tracking in post-processing. Furthermore, we implement a lens blur CUDA layer featuring a novel differentiable disk kernel that accurately simulates the realistic fall-off boundary of the PSF.

**Limitations.** Similar to the original layered neural atlases [Kasten et al. 2021], our model could not handle videos with objects in large self-occlusion, which require more layers of atlases. Notably, our method excels when objects experience many distinct defocus blur levels. However, performance worsens when fewer defocus levels are present, as illustrated in Fig. 8. The inset is heavily defocused and then occluded by the person, we can restore it but to a limited extent. This is attributed to the high degrees of freedom inherent in our problem.

ACKNOWLEDGMENTS

We would like to thank Dominique Piché-Meunier for helping with the evaluation of their method [Piché-Meunier et al. 2023] on our dataset and for the valuable comments from anonymous reviewers.

REFERENCES


Figure 9: We compare our method to relate work on real captured video (up) and rendered animation (bottom). We crop and enlarge the small inserts for better comparison with region locations indicated by blue and orange dash line rectangle. We slice two pieces of data (pink and yellow dash line) on time dimension and visualize at the last two columns for temporal consistency comparison. Blue arrows have been used to highlight the artifacts and temporal inconsistency. Agent 327 © 2024 Blender