



N-SfC: Robust and Fast Shape Estimation from Caustic Images — Supplementary Material —

Marc Kassubeck , Moritz Kappel , Susana Castillo , and Marcus Magnor 

Institut für Computergraphik, TU Braunschweig, Germany
{kassubeck, kappel, castillo, magnor}@cg.cs.tu-bs.de

In this supplementary document, we provide more details on the proposed methodology given in the main paper and implementation details, along with further comparisons on the methods examined in the main paper, additional results, and ablation studies for the presented framework.

We will also provide the full code including dataset generation scripts upon publication of this paper.

1. Method and Network Architecture

As mentioned in the paper, the simulation of the caustic image (cf. [FSES14]) and subsequent gradient calculation through back-propagation mainly depends on three hyperparameters: The number of samples n_j ; the number of distinct wavelengths associated with each ray n_w ; and the smoothing parameter s , which represents a trade-off between the minimum feature size, which can occur in the caustic image and the sampling noise of the simulation. An overview of further parameters and typical values can be found in Tab. 1.

Subsequent processing of caustic images is dependent on two neural-network components, which belong to the same architectural family, as depicted in Fig. 1. As mentioned in the main paper, both networks share a structure similar to UNet [RFB15] as the main component, and differ in a few blocks with respect to the input and output. The denoiser includes a single `Conv + nonlin` block, which expands the number of channels to $c_{init} \in [1, 32]$ channels for the UNet part of the network. An equivalent block contracts those channels after the UNet part of the denoising network. Note that the number of input channels in the denoising network is given as 1, even though the caustic images are multi-spectral with n_w channels. This means each channel is denoised independently, such that this network is trained to handle simulation input with as few as one or many spectral channels. Overall we consider a family of networks for both the denoising and the update parts, which are parameterized via the hyperparameters listed in Tab. 2. At training time we search for the best network architecture over the parameter space defined therein.

2. Datasets

Given our focus on the application of quality control in glass 3D printing, all the physical values for the non-differentiable scene

Table 1: Non-Differentiable Scene Parameters. Note that *non-differentiable* does not indicate an intrinsic limitation of the approach, but is meant to indicate that we do not optimize for these parameters and thus chose not to compute gradients for them. The last column denotes the values considered for the scene setup in dataset creation and the final optimization loops.

Parameter	Meaning	Value
$n \times n$	Pixel resolution of height map	(128, 128)
$m \times m$	Pixel resolution of caustic image	(512, 512)
$d \in \mathbf{R}$	Base thickness of substrate	3mm
$n_j \in \mathbf{N}$	Number of samples, <i>i.e.</i> light paths	$1e^6$ and $16e^6$
$n_w \in \mathbf{N}$	Number of wavelengths in simulation	3: {610nm, 530nm, 430nm}
$s \in \mathbf{R}$	Smoothing parameter for photon footprint	16.0
$\alpha \in \mathbf{R}_+$	Angle of light emission	0 (collimated light)
$L_i \in \mathbf{R}_+^{n_w}$	Radiosity of light source	(1 W/m ² , 1 W/m ² , 1 W/m ²)
$L_p \in \mathbf{R}_+^3$	Position of light source	(0m, 0m, 1m)
$S_p \in \mathbf{R}^3$	Position of screen	(0m, 0m, $-1e^{-6}$ m)
$(h_x, h_y) \in \mathbf{R}_+^2$	Size of the base substrate	(5cm, 5cm)

parameters were selected to match usual process values with current technology. As mentioned in the main paper, it is crucial that the training data accurately samples the distribution of real world values. Thus, to define parameters for our dataset of glass substrates it is beneficial to recall some parameters of the underlying production process. The process works by depositing glass fibers of diameter of roughly 0.4mm onto 3mm thick glass substrates, for which an area of 5×5 cm is common. In our simplified setup, only the source and the screen surface interact with the light paths. All parameters regarding scene structure can be found in Tab. 1. The parameters regarding sample count and resolution were chosen as a compromise between quality and memory consumption. For more details about the influence of each rendering parameter we refer to the work of Frisvad *et al.* [FSES14].

On the same line, our *denoising* and *updater* datasets are both rendered by drawing samples from distributions carefully chosen to closely match the real data distribution. In particular, the line samples are uniformly drawn from the ranges defined in Tab. 3 and have a cosine falloff from their center to the edge of their width, representing a fully fused glass fiber on top of the base substrate. To get the final heightfield we add these line samples to simulate the effect of printing a fiber on top of pre-existing ones.

The *denoising dataset* was created on a machine with a Xeon-E5

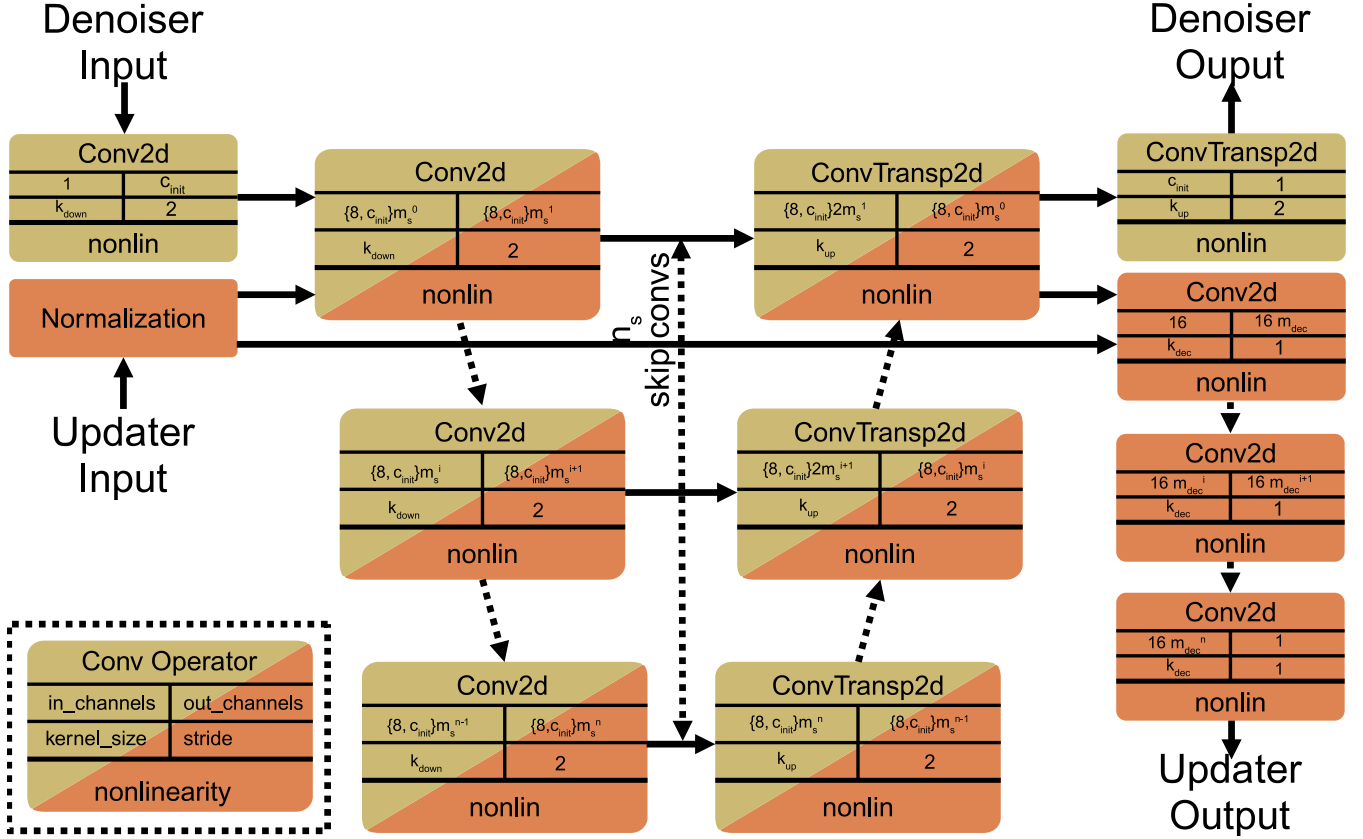


Figure 1: The network architectures are variants of the UNet [RFB15] architecture. The denoiser and updater network differ mainly in their input and output computations. The coloring depicts, whether a block belongs to the code-path of denoiser or updater respectively and when two inputs lead into a block, we concatenate along the channel dimension.

Table 2: Parameters for family of networks. Also see Fig. 1 for a graphical depiction for the family of considered network architectures.

Parameter	Meaning	Type	Search Value	Best Denoiser	Best Updater
learning rate	Learning rate of optimizer	float	$\in [0.0001, 0.1]$	0.00151	0.005155
c_{init}	Initial number of channels for UNet	integer	$\in [1, 32]$	31	-
$nonlin$	Specific nonlinearity to employ after convolutions	-	ELU, ReLU, PReLU, SELU	PReLU	PReLU
k_{down}	Kernel size in downconvolution part of network	integer	$\in [2, 11]$	5	9
k_{up}	Kernel size in upconvolution part of network	integer	$\in [2, 11]$	2	9
m_s	Channel multiplier for each depth layer of UNet	integer	$\in [1, 8]$	2	8
n_s	Number of skip connections; depth of UNet	integer	$\in [1, 4]$	4	1
m_{dec}	Channel divisor in each output block	integer	$\in [2, 11]$	-	8
k_{dec}	Kernel size in each output block	integer	$\in \{2, 4, 8, 16\}$	-	4
λ_{TV}	Total variation weight in loss for updater training	float	$\in [5 \cdot 10^{-10}, 5 \cdot 10^{-9}]$	-	$9.81 \cdot 10^{-10}$

1630 CPU and two NVidia Titan RTX GPUs with 24Gb memory each, where it took roughly 17h. The generation of the 100000 samples for the *updater dataset* took about 30h on a Xeon-E5 1630 CPU and two NVidia RTX 3090 GPUs with 24Gb memory each.

2.1. Denoiser Training

We implemented the denoising component of the network using PyTorch Lightning [Fe19] and trained with this dataset, split randomly into 90% training and 10% validation images. The loss function for

this training was the MSE loss between estimated denoised caustic and the caustic image with $1.6 \cdot 10^7$ samples. We searched over the hyperparameters as defined in Tab. 2 by minimizing this loss over the validation set to find the best performing architecture. This search was performed using the hyperparameter sweeps function in [Bie20] with their implementation of Bayesian search [SLA12] and hyperband early termination [LJD*17] with a minimum of 5 epochs. All models were trained with Adam optimizer [KB15] until the validation loss showed no improvement over the last 3 epochs. The total compute time of the search was 5 days with 60 trained

Table 3: Common Parameter ranges for heightfield generation and offsets for the updater dataset

Parameter	Primary Sample Range	Perturbation Offsets
Number of lines	{2, ..., 30}	None
Line Start	[-2.5cm, 2.5cm] ²	[-2.5mm, 2.5mm] ²
Line End	[-2.5cm, 2.5cm] ²	[-2.5mm, 2.5mm] ²
Line Width	[0.1mm, 4mm]	[-1mm, 1mm]
Line Height	[0.1mm, 2mm]	[-1mm, 1mm]

models over two machines. One being the afore mentioned Xeon-E5 1630 CPU and two NVidia Titan RTX GPUs with 24Gb memory and the other equipped with a Xeon-E5 1630 CPU and two NVidia RTX 3090 GPUs with 24Gb memory each. The parameters of the best performing model are listed in Tab. 2 and were used in all following experiments.

2.2. Updater Training

The reasoning behind the sampling for the *updater* dataset is that learned gradient descent schemes are usually trained in an unrolled fashion with multiple consecutive update steps being supervised by ground-truth data. Unfortunately it proved to be computationally infeasible to generate a new simulation and gradient after each update step while simultaneously optimizing for the best network architecture. Thus, we opted to create a larger dataset and train the updater point-wise by comparing the resulting height field to the target height field after a single step. We hypothesized that this dataset contains enough varied image patches to let the network learn the dynamics of a learned gradient descent scheme and generalize to repeated execution at test time. The rest of the training framework like dataset split, computational setup, etc., is the same as for the denoising network. Total compute time was 20 days with 89 trained models, with the best parameters as reported in Tab. 2.

3. Comparisons

We provide complete renderings of the full test set containing the ten heightfields considered in the comparisons and evaluation of our method mentioned on the main paper, which have been modified for better visibility in Fig. 2. Along with these ground truth renderings, we also provide the corresponding reconstructions of SfC [KBC*21] in Fig. 3 and our full model in Fig. 5. It is clear that SfC struggles with the high-frequency components in the caustics, resulting in narrow and steep reconstructions. Further, we provide renderings of the reconstructions of HCCCD [STTP14] in Fig. 4. Please note that this method is prone to changing the thickness of the base substrate, which is not very visible in Fig. 4, but better appreciated in the numerical evaluation in the paper.

4. Ablations

We provide renderings of our ablations, *i.e.* the updater trained without the denoiser in place and the updater without the gradient information from the differentiable renderer in Fig. 6 and Fig. 7. Comparing these reveals, that little visual difference is present in these model variants at the displayed steps, but further numerical

analysis in the paper reveals differences in the convergence dynamics between them.

References

- [Bie20] BIEWALD L.: Experiment tracking with weights and biases, 2020. Software available from wandb.com. URL: <https://www.wandb.com/>. 2
- [Fe19] FALCON W., *et al.*: Pytorch lightning, 2019. Software available from pytorchlightning.ai. 2
- [FSES14] FRISVAD J., SCHJØTH L., ERLEBEN K., SPORRING J.: Photon differential splatting for rendering caustics. *Comput. Graph. Forum* 33, 6 (2014), 252–263. 1
- [KB15] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. In *Int. Conf. Learn. Represent.* (2015). 2
- [KBC*21] KASSUBECK M., BÜRCEL F., CASTILLO S., STILLER S., MAGNOR M.: Shape from caustics: Reconstruction of 3D-printed glass from simulated caustic images. In *IEEE Winter Conf. Appl. Comput. Vis.* (2021), pp. 2877–2886. 3, 4
- [LJD*17] LI L., JAMIESON K., DESALVO G., ROSTAMIZADEH A., TALWALKAR A.: Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research* 18, 1 (2017), 6765–6816. 2
- [RFB15] RONNEBERGER O., FISCHER P., BROX T.: U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)* (2015), Springer, pp. 234–241. 1, 2
- [SLA12] SNOEK J., LAROCHELLE H., ADAMS R. P.: Practical bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems* 25 (2012). 2
- [STTP14] SCHWARTZBURG Y., TESTUZ R., TAGLIASACCHI A., PAULY M.: High-contrast computational caustic design. *ACM Trans. Graph.* 33, 4 (2014). 3, 4

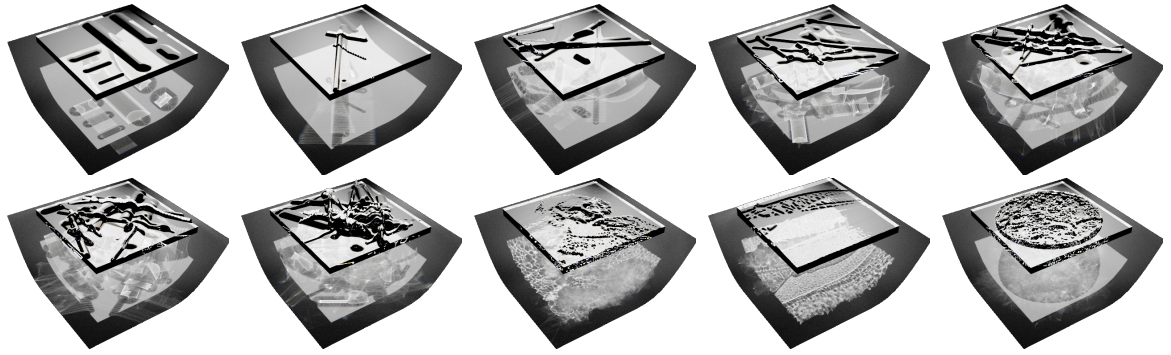


Figure 2: *Ground truth renderings of the test set in a modified scene for better visibility. Please note that optimization assumes a flat screen surface and significantly less distance between substrate and screen.*

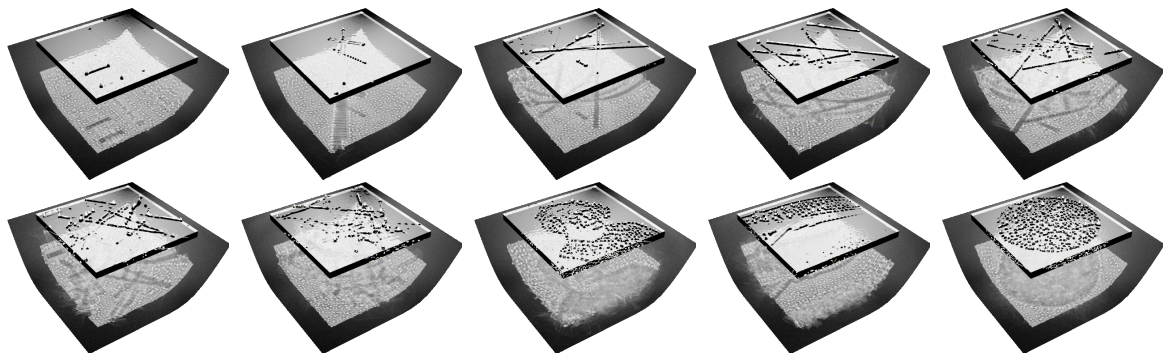


Figure 3: *Renderings of SfC [KBC*21] reconstructions after one update iteration.*

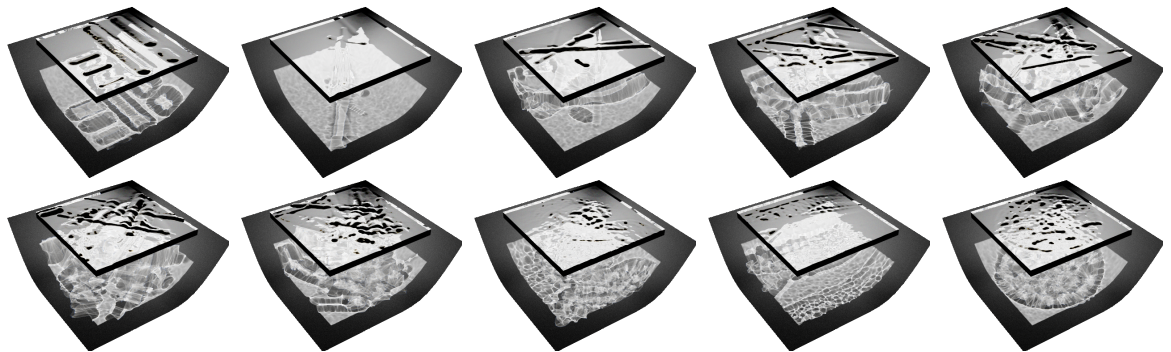


Figure 4: *Renderings of HCCCD [STP14] reconstructions.*

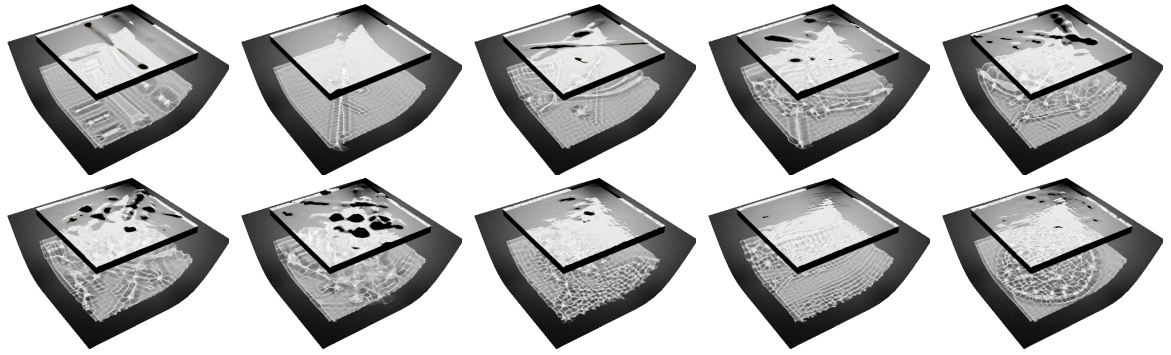


Figure 5: *Renderings of N-SfC reconstructions after ten update iterations.*

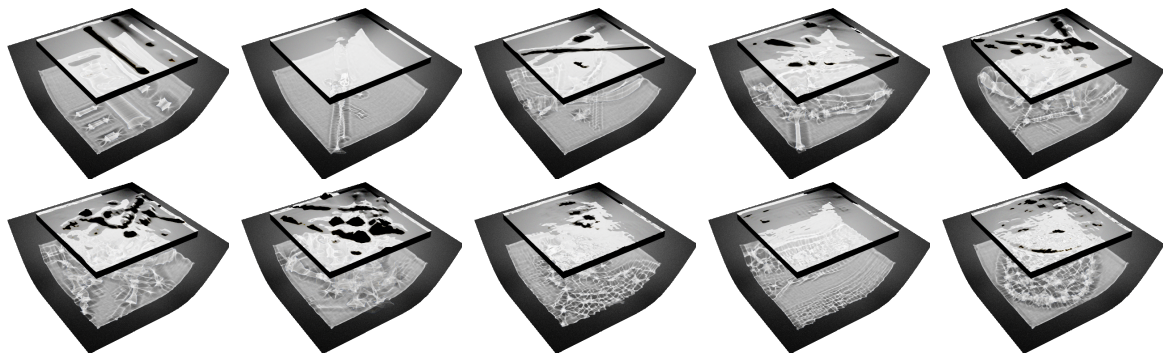


Figure 6: *Renderings of N-SfC reconstructions without the denoiser after ten update iterations.*

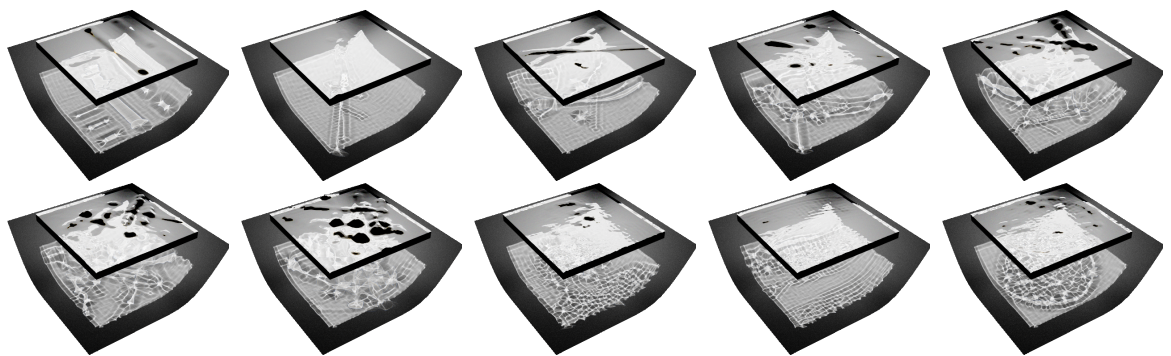


Figure 7: *Renderings of N-SfC reconstructions without the local gradient information after ten update iterations.*