A. Gradient derivation

The i-th element of the gradient of the Gaussian, is

$$\nabla_i \mathcal{N}(\boldsymbol{\tau}) = \frac{\partial_i \mathcal{N}(\boldsymbol{\tau})}{\partial \tau_i} \tag{24}$$

$$= \frac{\mathrm{d}\mathcal{N}(\tau_i)}{\mathrm{d}\tau_i} \prod_{\substack{j=1\\ j\neq i}}^{n} \mathcal{N}(\tau_j) \tag{25}$$

$$= -\frac{\tau_i}{\sigma^2}\mathcal{N}(\tau_i) \prod_{\substack{j=1\\ j\neq i}}^{n} \mathcal{N}(\tau_j) \tag{26}$$

$$= -\frac{\tau_i}{\sigma^2}\mathcal{N}(\boldsymbol{\tau}), \tag{27}$$

where we use the overloaded convention that $\mathcal{N}(\boldsymbol{\tau})$ takes a vector $\boldsymbol{\tau}$ and $\mathcal{N}(\tau_i)$ the i-th element, a scalar τ_i, to produce the one or n-dimensional Gaussian.

This differs from Fischer and Ritschel [7] who only blur in the direction in which they differentiate, as in

$$\nabla_i \mathcal{N}(\tau_i) = \frac{\tau_i}{\sigma^2}\mathcal{N}(\tau_i), \tag{28}$$

while it is more consistent with higher-order differentials to blur all dimensions.

B. Smooth Gradient Marginalization

The sampling of 1D Gaussian gradient was derived by Fischer and Ritschel [7] who constructed a PDF p^{G}. For a n-dimensional Gaussian gradient, we need to marginalize and sample the dimensions individually. Marginalization would be integration over all dimensions $j \neq i$, except the one we look for i, so:

$$\int_{\boldsymbol{\tau}_{j\neq i}} p^{\mathrm{G}}(\boldsymbol{\tau})\mathrm{d}\boldsymbol{\tau}_{j\neq i}. \tag{29}$$

Writing out the CDF, a positivized and scaled version of the PDF p, where $1/Z$ is the partition function, gives

$$\frac{1}{Z}\int_{\boldsymbol{\tau}_{j\neq i}} \frac{1}{2}|\nabla_i \mathcal{N}(\boldsymbol{\tau})|\mathrm{d}\boldsymbol{\tau}_{j\neq i} = \tag{30}$$

$$\frac{1}{Z}\int_{\boldsymbol{\tau}_{j\neq i}} |\nabla_i \mathcal{N}(\boldsymbol{\tau})|\mathrm{d}\boldsymbol{\tau}_{j\neq i}. \tag{31}$$

Writing the n-D Gaussian as product of n 1D Gaussians

$$\frac{1}{Z}\int_{\boldsymbol{\tau}_{j\neq i}} |\nabla_i \prod_{j=1}^{N} \mathcal{N}(\tau_j)|\mathrm{d}\tau_{j\neq i}. \tag{32}$$

As we differentiate only by τ_i, all other factors are 1, so

$$\frac{1}{Z}\int_{\tau_{j\neq i}} |\nabla_i \mathcal{N}(\tau_i)|\mathrm{d}\tau_{j\neq i}. \tag{33}$$

As we are integrating over all $\tau_{j\neq 1}$, integration becomes multiplication with the domain's measure, 1.

$$|\nabla_i \mathcal{N}(\tau_i)|. \tag{34}$$

C. Hessian derivation

The diagonal elements of the Hessian of the Gaussian are

$$\nabla^2 \mathcal{N}(\boldsymbol{\tau})_{ii} = \frac{\partial_i^2 \mathcal{N}(\boldsymbol{\tau})}{\partial^2 \tau_i} \tag{35}$$

$$= \frac{\partial}{\partial \tau_i}\left(-\frac{\tau_i}{\sigma^2}\mathcal{N}(\tau_i) \prod_{\substack{j=1\\ j\neq i}}^{n} \mathcal{N}(\tau_j)\right) \tag{36}$$

$$= \frac{\partial}{\partial \tau_i}\left(-\frac{\tau_i}{\sigma^2}\mathcal{N}(\tau_i)\right) \prod_{\substack{j=1\\ j\neq i}}^{n} \mathcal{N}(\tau_j) \tag{37}$$

$$= \left(-\frac{1}{\sigma^2} + \frac{\tau_i^2}{\sigma^4}\right) \prod_{\substack{j=1\\ j\neq i}}^{n} \mathcal{N}(\tau_j) \tag{38}$$

$$= \left(-\frac{1}{\sigma^2} + \frac{\tau_i^2}{\sigma^4}\right) \mathcal{N}(\boldsymbol{\tau}). \tag{39}$$

The non-diagonals of the Hessian of the Gaussian are

$$\nabla^2 \mathcal{N}(\boldsymbol{\tau})_{ij} = \frac{\partial^2 \mathcal{N}(\boldsymbol{\tau})}{\partial_i \tau_i \partial_j \tau_j} \tag{40}$$

$$= \frac{\partial}{\partial \tau_j}\left(-\frac{\tau_i}{\sigma^2}\mathcal{N}(\tau_i) \prod_{\substack{k=1\\ k\neq i}}^{n} \mathcal{N}(\boldsymbol{\tau}_k)\right) \tag{41}$$

$$= \frac{\tau_i}{\sigma^2}\mathcal{N}(\tau_i)\frac{\tau_j}{\sigma^2}\mathcal{N}(\tau_j) \prod_{\substack{k=1\\ k\neq i,j}}^{n} \mathcal{N}(\boldsymbol{\tau}_k) \tag{42}$$

$$= \frac{\tau_i}{\sigma^2}\frac{\tau_j}{\sigma^2}\mathcal{N}(\tau_i)\mathcal{N}(\tau_j) \prod_{\substack{k=1\\ k\neq i,j}}^{n} \mathcal{N}(\boldsymbol{\tau}_k) \tag{43}$$

$$= \frac{\tau_i \tau_j}{\sigma^4}\mathcal{N}(\boldsymbol{\tau}). \tag{44}$$

A remark: It might appear, that diagonal is a special case of off-diagonal, but for differentiation, that is not true, as on the diagonal, the variable we differentiate in respect to appears twice, in the sense that $\mathrm{d}uv/\mathrm{d}u = v$ and $\mathrm{d}uv/\mathrm{d}v = u$, but $\mathrm{d}uu/\mathrm{d}u = 2u$.

D. Sampling diagonal of Hessian

Similar to Sec. B, for the diagonal of the Hessian, we can sample each dimension independently. Thus, we can first derive the valid distribution of the second-order derivative of the one-dimensional Gaussian by positivization and scaling, and it will apply to higher dimensions: The one-dimensional Gaussian's second-order derivative is

$$\left(-\frac{1}{\sigma^2} + \frac{\tau_i^2}{\sigma^4}\right) \mathcal{N}(\tau_i) \tag{45}$$

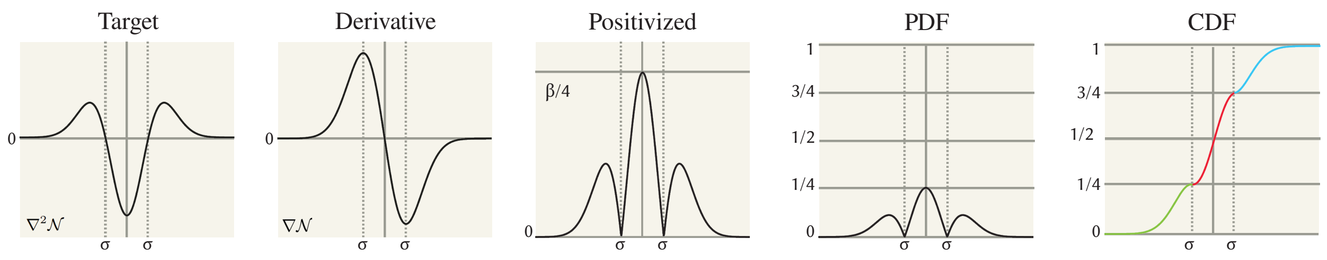

Figure 6. Detailed plots of the functions involved in the derivation of the CDF for the diagonal elements of $\nabla^2\mathcal{N}$, extending Fig. 3.

The roots of Eq. 45 are the τ_i for which

$$\left(-\frac{1}{\sigma^2}+\frac{\tau_i^2}{\sigma^4}\right)\mathcal{N}(\tau_i)=0. \tag{46}$$

As $\mathcal{N}(\tau_i)>0$ for all τ_i, the product can be 0 only if

$$-\frac{1}{\sigma^2}+\frac{\tau_i^2}{\sigma^4}=0 \quad \text{and hence} \tag{47}$$

$$\tau_i=\pm\sigma. \tag{48}$$

The function value between $-\sigma<\tau_i\leq\sigma$ is negative and hence needs to be positivised. Since the second-order derivative should integrate to the gradient of the Gaussian, we know that it reaches zero as τ_i reaches infinity. In conjunction with the fact that the second-order derivative is symmetric about the y-axis, we can conclude that the integral of the interval $-\sigma<\tau_i\leq\sigma$ should be twice the size of the integral of $\tau_i\leq-\sigma=\tau_i>\sigma$. Thus, after positivization, the CDF should be scaled, such that it is $\frac{1}{4}$ at $\tau_i=-\sigma$. Solving for these equalities, we can get:

$$\beta\nabla\mathcal{N}(-\sigma)=\frac{1}{4} \tag{49}$$

$$\beta=\frac{1}{4\nabla\mathcal{N}(-\sigma)}. \tag{50}$$

So, for the positivised rescaled second-order derivative as a PDF of the distribution:

$$p^{\mathrm{H}}_{ii}=|\beta\nabla^2\mathcal{N}(\tau_i)|. \tag{51}$$

We can get the integrating constant by flipping and translating the scaled gradient of Gaussian to arrive at the CDF function for the intervals:

$$P^{\mathrm{H}}_{ii}(\tau_i)=\begin{cases}\beta\nabla\mathcal{N}(\tau_i) & \text{if } \tau_i<-\sigma,\\ \frac{1}{2}+\beta\nabla\mathcal{N}(\tau_i) & \text{if } \tau_i\in[-\sigma,\sigma]\\ 1-\beta\nabla\mathcal{N}(\tau_i) & \text{if } \tau_i>\sigma.\end{cases} \tag{52}$$

E. Grey-box differentials

Sometimes, differentials are in respect to a function that is a composition $\mathbf{z}=f(\mathbf{y}=g(\mathbf{x}))$ of an inner function with known analytic differentials g (white box) and an outer function f with differentials that need to be sampled (black box). For first order (gradient), this is

$$\nabla_{\mathbf{z}}(\mathbf{z}=f(g(\mathbf{x})))=(\nabla_{\mathbf{x}}g(\mathbf{x}))^{\mathsf{T}}\cdot\nabla_{\mathbf{y}}f(\mathbf{y}=g(\mathbf{x})),$$

which means to take the Jacobian (as both g and f in general are vector-valued) of the inner function g in respect to the inner argument $\mathbf{x}$ and vector-matrix multiply this with the gradient of the outer function f but in respect to the outer argument $\mathbf{y}$. For the second order it is

$$\nabla^2_{\mathbf{z}}f(g(\mathbf{x}))\approx(\nabla_{\mathbf{x}}g(\mathbf{x}))^{\mathsf{T}}\cdot\nabla^2_{\mathbf{y}}f(g(\mathbf{x}))\cdot\nabla_{\mathbf{x}}g(\mathbf{x})$$

which means again to take the gradient of the inner function, but multiply it with the Hessian, instead of the Jacobian of the composition in respect to the outer argument [17].

The aim of this exercise is to have the sampled gradients handle only the black-box part and the analytic gradients handle the non-sampled parts. As the analytic parts are typically large (*e.g.*, in the order of the size of a neural network) compared to the number of physical rendering parameters (placement of light, cameras or objects), this can provide substantial advantages.

F. BFGS/LBFGS method

Quasi-Newton methods are also a way to utilize the second order information for optimization, however, they approximate this information with zero or first order information. We tested the family of algorithms from the quasi-Newton methods that is known to be most effective, the BFGS algorithms [30]. For this family of algorithms, the vanilla BFGS algorithm [29], along with BFGS with Armijo-Wolfe line search [22], and LBFGS [22], were tested, but they only converge for the QUAD task. This is probably due to the noisy nature of the derivative estimation. To this end, damped BFGS [25] and adaptive finite difference BFGS [1] were also added, but neither changed the convergence of other tasks.

G. Comparing to analytical Hessians

To further validate our approach, we investigate a task where the analytic Hessians are available: the optimization of a sphere's Phong BRDF (7 unknowns) under point illumination. We use this task to evaluate both analytical and sampled derivatives, as well as the corresponding first and second-order methods that employ them in Fig. 7. The dashed lines indicate the first-order methods, while the solid lines represent the second-order methods. We see that

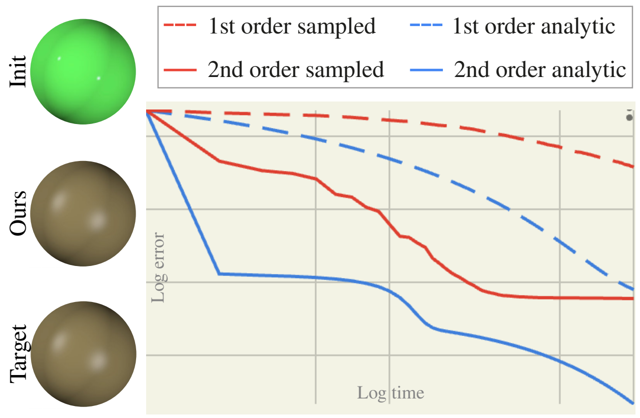

Figure 7. BRDF optimization.

the sampled method is less accurate and takes longer due to the sample size and the bandwidth of sampling. The second-order methods outperform the first-order ones in both respects. We show the outcome of our second-order optimization denoted as 'Ours' in Fig. 7.

H. Positive-Definite Hessians

Naive Newton will fail for cases where the problem's Hessian is not positive semi-definite (PSD). For a non-positive definite Hessian, there may exist a unbounded, negative eigenvalue. This means that when the Hessian is applied to a gradient vector, it may flip the gradient vector and cause opposite divergence from the gradient's descent direction. Although optimization is separate from our contribution of stochastic gradient estimation, we demonstrate how our method deals with this issue with an example. To this end, consider a negated 2D Gaussian $-\mathcal{N}(0, \sigma_1)$ whose Hessian is non-PSD everywhere. Its convolution with a second Gaussian $-\mathcal{N}(0, \sigma_2)$ results in a third Gaussian $-\mathcal{N}(0, \sigma_3)$, whose Hessian is also not PSD everywhere. In this example, we know the analytic expressions of all the Gaussians, so we can compute their Hessians and compare them to our estimates. The mean error of our method across the interval $(-3, 3)^2$ is within 1×10^{-4} for 100 samples. With Hessian modification and trust region, running an optimization finds the correct minimum at (0,0) for any starting point in $(-3, 3)^2$. Fig. 8 shows the optimization error across an ensemble of 20 runs in addition to a top-view of the optimization trajectories for both the analytic and sampled derivatives. This indicates that, for the right combination of estimator and optimizer, non-PSD is not necessarily a problem.

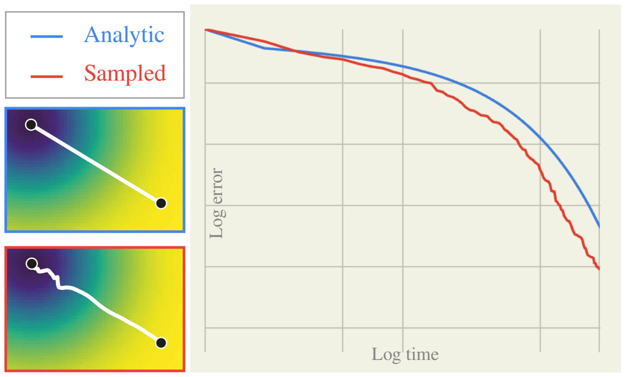

Figure 8. Optimization on a problem with non-PSD Hessians.

I. Robustness analysis

We investigate our method's robustness to:

Initialization. All our experiments are performed over an ensemble of 20 random starting points. The reported numbers and findings are the average of these optimization outcomes.

Hessian sample count. We repeated the MUG task with different numbers of samples used to estimate the Hessian during optimization (using multiples of two, due to antithetic sampling). Our overall findings are consistent with prior work and indicate that increasing the number of estimation samples does help convergence. However, the increase timed for higher sample counts is not offset by improved convergence for this task, as the per-iteration convergence gain does not sufficiently compensate for the slowdown (left subplot in Fig. 10). This indicates that the minimal number of antithetic samples (two) can be optimal for a relevant task.

Rendering MC noise. We have repeated the MUG experiment with 0.25×, 0.5×, 1×, 2×, and 3× the number of rendering samples and did not observe significant change in optimization quality and convergence. However, as the sample count for rendering increased, the time taken to render each image is longer, leading to a slower convergence rate for higher sample counts (center subplot in Fig. 10). This provides us with a data point that noise from low-spp MC rendering is not a dominating limiting factor for our method, although a more in-depth investigation would be needed to reliably confirm this across all experiments.

Real-world compression noise. We repeated the MUG task with JPEG compression at the 2%, 4%, 6%, 9%, and 14% levels and did not observe significant degradation in convergence (right subplot in Fig. 10). The difference in the final result is caused by the loss calculation between the rendered image and the noisy, compressed image.

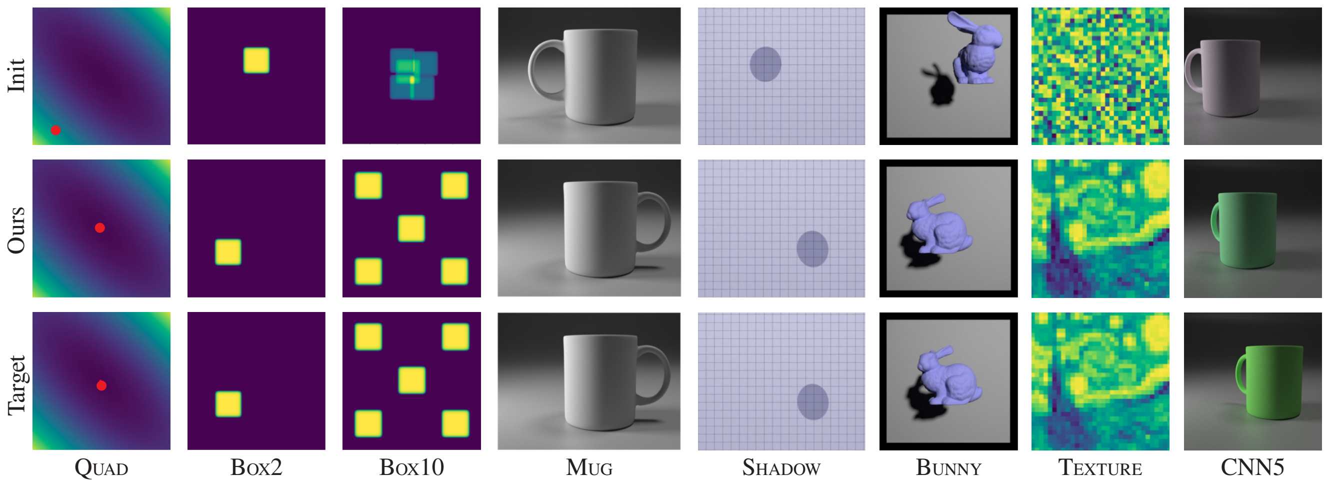

Figure 9. Overview of all tasks we study (columns): the first row shows the task initialization, the starting point of the optimization. The middle row shows the outcome of optimizing with our estimated Hessians with `OursHVPA`, while the bottom row shows the ground truth for each task.

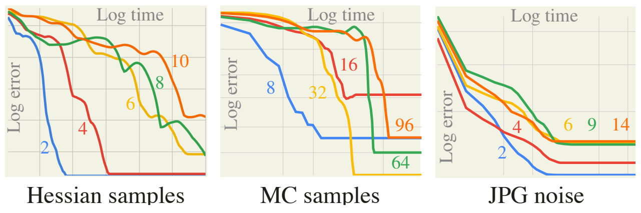

Figure 10. Robustness evaluation (see text). All plots are displayed on a log-error vs. log-time scale.

J. How much variance is reduced?

In Fig. 11, we perform a variance analysis of the different estimators on different differentiable quantities. We see that all estimators converge linearly in a log-log plot. The optimal estimator (dotted) has the lowest variance and hence would lead to the least noise in optimization, but at the expense of evaluating quadratically many elements for Hessians. Our aggregate sampling (solid line) performs slightly worse, but much, better than uniform sampling (thin line) would do.

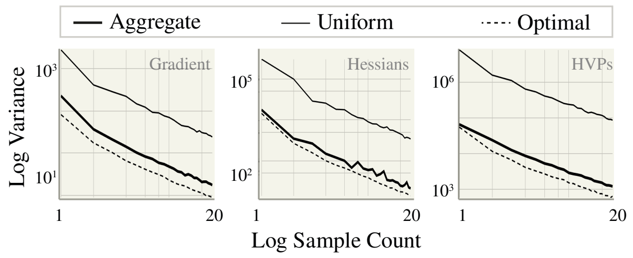

Figure 11. Estimator variance (lines) for different operators (plots).

K. Additional Experiments

We perform two additional experiments to show that our method can successfully differentiate diverse light transport scenarios: in Fig. 12 we have repeated the caustic example from ZeroGrads [8], where the goal is to optimize a height-field, parametrized by a 1,024-dimensional B-spline, such that the caustic it creates matches a reference image. Our method performs well on this task, even thought the loss landscape is highly non-convex and the optimization variables exhibit highly non-local image-space interactions.

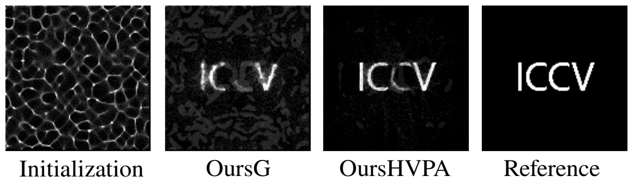

Figure 12. Inverse optimization of a heightfield such that the caustic it creates when light shines through it matches a reference. CMA-ES does not converge on this task.

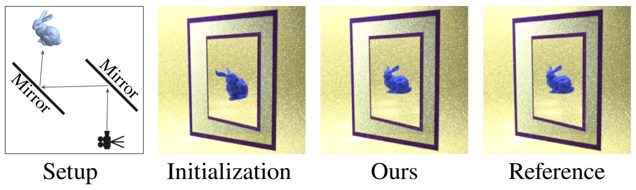

Figure 13. Two-bounce mirror optimization experiment.

Additionally, we optimize a two-bounce experiment, where the rotation (around the up-axis) and translation (along x,y) of the Stanford bunny are optimized, while the bunny is only observed through two mirrors. The left subfigure in Fig. 13 shows the experiment. We use our method `OursHVPA` and observe a successful optimization outcome.

Method	Parameter	QUAD	BOX2	BOX10	MUG	SHAD	BUNNY	TEXTURE	CNN	CNN5
	Samples	4	6	6	1	1	2	4	1	1
	Sigma (start)	1	1.5	0.6	3	0.5	1	0.5	0.5	0.5
`FR22`	Learning rate	0.5	0.3	0.05	0.1	0.02	0.02	0.05	1e-4	1e-4
	Sigma (end)	0.01	0.01	0.1	0.01	0.01	0.01	0.1	0.01	0.01
`OurG`	Learning rate	0.5	0.3	0.05	0.1	0.02	0.02	0.05	1e-4	1e-4
	Sigma (end)	0.01	0.01	0.1	0.01	0.01	0.01	0.1	0.01	0.01
`OurH`	Trust region	50	2	10	3	3	2	10	1e-2	1e-2
	Sigma (end)	0.01	0.01	0.1	0.01	0.01	0.005	0.1	0.01	0.01
	Line search iteration	5	10	3	1	1	1	5	2	2
	Line search tolerance	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3
	Recompute	5	10	30	10	10	5	30	20	20
`OurHVP`	Trust region	50	2	10	4	5	4	10	1e-2	1e-2
	Sigma (end)	0.05	0.01	0.1	0.01	0.01	0.01	0.1	0.01	0.01
	Line search iteration	5	3	3	2	1	1	5	2	2
	Line search tolerance	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3
	Recompute	5	10	30	2	5	10	30	20	20
`OurHVPA`	Trust region	50	2	10	4	3	4	10	1e-2	1e-2
	Sigma (end)	0.05	0.01	0.1	0.01	0.01	0.01	0.1	0.01	0.01
	Line search iteration	5	1 0	3	1	10	2	5	2	2
	Line search tolerance	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3
	Recompute	5	10	30	5	5	2	30	20	20

Table 3. Hyperparameters for our methods (rows) on the different tasks (columns). All parameters, including those of our competitors, have been optimally chosen.

L. Hyperparameters

In this section, we detail the hyperparameters of our experiments and show the initial-, output- and ground-truth images for each task. The hyperparameters are shown in Tab. 3, and the task images are shown in Fig. 9. The sample size, which is always antithetic and doubles the sample size, applies to our methods and FR22. All Sigma annealing, which controls the bandwidth σ of the distribution [7], is scheduled linearly and has a start and end value. For first-order methods, the tunable hyperparameter is the learning rate. For second-order ones, the trust region value shows the initial search bound, and recompute is the number of iterations until the line search is re-estimated.