

DISK: DIFFERENTIABLE SPARSE KERNEL COMPLEX FOR EFFICIENT SPATIALLY-VARIANT CONVOLUTION

Zhizhen Wu* Zhe Cao* Yuchi Huo†

State Key Lab of CAD&CG, Zhejiang University, China

zhizhenwu@zju.edu.cn, caozhe022@qq.com, huo.yuchi.sc@gmail.com

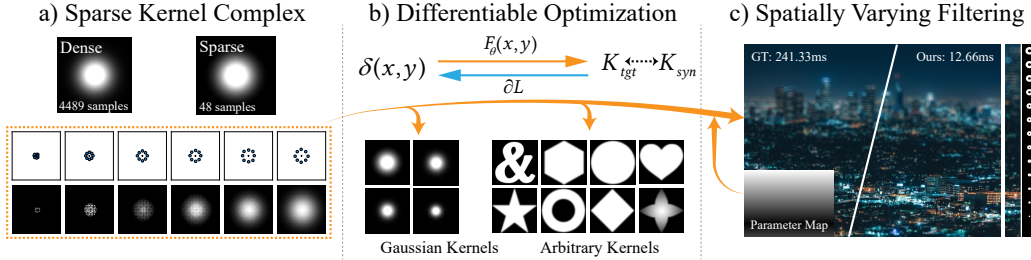


Figure 1: **An overview of our method.** We represent a dense filter as a *Sparse Kernel Complex*, a sequence of sparse layers whose parameters Θ are learned via differentiable optimization. We apply our filter F_{Θ} to an impulse δ to yield a synthesized kernel K_{syn} , and minimize a loss $\mathcal{L}$ against the target K_{tgt} to learn arbitrary shapes. These optimized kernels form a basis for high-performance spatially varying filtering, achieving quality close to ground truth with up to a $20\times$ speedup.

ABSTRACT

Image convolution with complex kernels is common in photography, scientific imaging, and animation, but dense convolution is too expensive for resource-limited devices. Existing approximations, such as simulated annealing and low-rank decompositions, are either slow or struggle with non-convex kernels. We present a differentiable kernel decomposition framework that represents a spatially variant dense kernel with a small set of sparse samples, assuming the target dense kernel is known for both optimization and filtering. Our method provides (i) end-to-end differentiable sparse-kernel optimization, (ii) shape-aware initialization for non-convex kernels, and (iii) kernel-space interpolation for efficient, multi-dimensional spatially varying filtering without retraining or added runtime cost. Across Gaussian and non-convex kernels, our method achieves higher fidelity than simulated annealing and lower cost than low-rank decomposition. It is practical for mobile imaging and real-time rendering, and integrates cleanly into learning pipelines.

1 INTRODUCTION

From rendering realistic depth-of-field effects (Sakurikar; Wu et al., 2022) in computational photography to modeling the intricate point spread functions (Liu et al., 2022; Shajkofci & Liebling, 2020) of optical systems, the ability to apply large, complex convolution kernels is a fundamental building block in modern vision and graphics computing systems. This creates a fundamental tension: while larger, more intricate kernels enable higher-fidelity results, their quadratic computational cost renders direct implementation impractical for interactive applications on devices ranging from mobile phones to high-end GPUs.

*Equal contribution.

†Corresponding author.

To bridge this gap, many works have focused on approximation strategies. For specific cases like Gaussian blur, elegant constant-time solutions (Zing, 2010; Kovesei, 2010) exploit the filter’s analytic structure. However, such specialized methods are not applicable to the arbitrary, often non-convex kernels needed for advanced effects. More general approaches, such as low-rank matrix decomposition (McGraw, 2015), support arbitrary kernels but typically reduce the computation to a sequence of smaller dense convolutions, which limits sparsity and caps efficiency gains.

A more direct and efficient approach (Schuster et al., 2020) is to approximate a dense kernel with a truly sparse one, drastically reducing the number of required computations. It relies on heuristic search via parallel simulated annealing to identify sparse sampling patterns for arbitrary kernels. Despite its generality, it requires many iterations and still misses high-fidelity solutions due to the non-convex optimization landscape. This motivates a more principled and efficient way to obtain high-quality sparse kernel representations.

In this work, we introduce a differentiable kernel decomposition framework to address this challenge. Like prior approximation methods, our formulation assumes access to the dense target kernel throughout optimization and filtering. Unlike traditional low-rank methods, we optimize a sequence of natively sparse kernels, yielding an efficient representation for runtime inference. We cast the decomposition as an end-to-end optimization problem and apply gradient-based methods instead of heuristic search such as simulated annealing, enabling more reliable convergence to high-fidelity solutions in far fewer iterations. To improve stability for non-convex target kernels, we further propose a two-part initialization strategy: structure-aware sampling to capture fine shape details and a deterministic radial initialization to stabilize and accelerate convergence.

Beyond single-kernel approximation, our framework supports efficient, multidimensional spatially varying (SV) filtering. The main challenge is the cost of generating a unique kernel per pixel, which often creates a significant performance bottleneck. We address this with filter-space interpolation: we precompute an optimized basis of sparse filters spanning the target effect range, then synthesize per-pixel kernels at runtime by interpolating this compact basis. This strategy reduces per-pixel synthesis to a minimal set of multiply-add operations, decoupling kernel-generation cost from image resolution and enabling sophisticated SV effects with negligible overhead.

Our contributions are as follows:

- A differentiable framework for decomposing a dense, arbitrary kernel into a sequence of optimized sparse layers, enabling efficient, high-fidelity approximation.
- An initialization scheme, combining a general radial strategy for stable convergence with a sparse sampling method for capturing non-convex kernels.
- A filter-space interpolation method for high-performance, spatially-varying filtering that decouples kernel synthesis cost from image resolution.

2 RELATED WORK

2.1 HIGH-PERFORMANCE KERNEL

Given that Gaussian blur is computationally expensive, numerous methods have been proposed to accelerate it. Common examples include $O(1)$ approximations such as the Extended Binomial Filter (Zing, 2010) and Summed-Area Table-based methods (Kovesei, 2010). However, their reliance on precomputation or inherently sequential steps makes them ill-suited to the massively parallel design of modern GPUs.

A better match for real-time rendering is Kawase blur (Kawase, 2003), a multi-pass filter that uses only four texture samples per pass. As an extension, Dual Filtering (Martin et al., 2015) introduces downsampling followed by upsampling, reducing bandwidth and computation by operating on lower-resolution textures. However, such pipelines trade regular downsampling for extra reconstruction work (often additional convolution) and may lose fine detail, whereas we stay at a single resolution and reduce cost via flexible sparse sampling while matching the target kernel response. A further practical limitation is the lack of a systematic mapping from a target Gaussian strength (e.g., a given σ) to the corresponding Kawase or Dual Filtering parameters; our work addresses this gap.

2.2 SPATIALLY-VARIANT FILTERING

A substantial body of work learns spatially varying, per-pixel convolution kernels, with applications to video prediction, frame interpolation, denoising, and deblurring (Jia et al., 2016; Niklaus et al., 2017; Mildenhall et al., 2018; Zhou et al., 2019; 2021). Unlike approaches that directly predict a dense per-pixel kernel map, our method decouples filter generation from spatial resolution. Specifically, we learn a compact lookup table (LUT) that parameterizes a continuous family of filters, enabling flexible and efficient spatially-variant filtering. Spatiotemporal Variance-Guided Filtering (Schied et al., 2017) forms per-pixel filter mixtures guided by estimated spatial and temporal variance. Differently, we condition filter generation on an input per-pixel blur-intensity map. This allows us to synthesize filters that match any target spatially varying blur, without intermediate variance estimation or other content statistics.

2.3 KERNEL APPROXIMATION AND DECOMPOSITION

Inspired by Kawase blur (Kawase, 2003), High-Performance Image Filters (Schuster et al., 2020) uses parallel tempering to optimize sparse sampling patterns, but its optimization stability can be limited by sensitivity to many hyperparameters compared with our gradient-based formulation. Kernel decomposition is also used to reduce cost in other settings: 2D kernels into pairs of 1D kernels for video interpolation (Niklaus et al., 2017); 3D spatiotemporal kernels into spatial and temporal atoms (STDCF) (Schied et al., 2017); standard convolution into depthwise and 1×1 pointwise operations (Howard et al., 2017; Chollet, 2017; Ramadhani et al., 2024); dynamic weights into static bases plus residuals (Li et al., 2021; 2024); and related decompositions for graph-transformer attention (KDLGT) (Wu et al., 2023) and large depthwise kernels (LKD-Net) (Luo et al., 2023).

Despite their success, these methods are largely limited to structured factorizations or heuristic optimization. Our framework addresses the missing capability: a differentiable, general solution for sparse approximation of arbitrary (including non-convex) kernels, together with an efficient mechanism for spatially varying kernel synthesis.

3 PRELIMINARY

3.1 KERNEL-BASED FILTERING

Kernel-based filtering is fundamental to many image processing tasks. This process takes an input image I_{in} and computes each pixel’s value for the output image I_{out} as a weighted average of its local neighbors within I_{in} . Formally, this operation is expressed as a 2D convolution, defined as:

$$I_{out}[x, y] = (I_{in} * K)[x, y] = \sum_{i=-k}^k \sum_{j=-k}^k I_{in}[x + i, y + j] \cdot K[i, j], \quad (1)$$

where the matrix K is the $M \times M$ convolution with kernel size $M \in \mathbb{R}^+$, whose elements $K[i, j]$ are weights that determine the contribution of each neighboring pixel to the final filtered value.

3.2 FILTER REPRESENTATION

The dense matrix representation for the kernel K in Eq. (1) is straightforward. However, its $O(M^2)$ computational cost presents a significant bottleneck. This is especially true for filters with a large spatial support, such as a Gaussian blur with a large σ , where the cost becomes prohibitively expensive for real-time applications that demand high frame rates.

Our key insight is to approximate this expensive operation by structuring the filter as a sequence of lightweight convolutional layers, where the output of one layer serves as the input for the subsequent one. Each layer applies a highly efficient sparse kernel, K_{sparse} , which we define by a small collection of N samples with offset-weight pairs:

$$K_{sparse} = \{(\mathbf{o}_i, w_i)\}_{i=1}^N, \quad (2)$$

where $\mathbf{o}_i \in \mathbb{R}^2$ is the spatial offset and w_i is its corresponding weight.

The complete operation, consisting of L such layers with kernels $(K_1, K_2, \dots, K_L)$, can be expressed as a nested convolution:

$$I_{out} = (\dots((I_{in} * K_1) * K_2) * \dots * K_L). \quad (3)$$

This multi-layer filter reduces the cost to $O(\sum_{l=1}^L N_l)$ per pixel. Since this sum is far smaller than the number of weights in the target dense kernel ($\sum N_l \ll M^2$), the approach offers a dramatic speedup.

4 METHODOLOGY

4.1 DIFFERENTIABLE MULTI-LAYER KERNEL COMPLEX

Overview Sparse filters offer a computationally efficient alternative to dense kernels; however, they often fail to capture the intricate structure of large, complex filters. The core challenge lies in determining the optimal parameters—the spatial offsets and weights—for a sequence of sparse kernels to accurately reconstruct a target. Manually designing these parameters or using traditional, non-differentiable methods is a formidable task.

To overcome this, our key contribution is to frame the decomposition as a differentiable optimization problem. This enables the simultaneous end-to-end learning of all sparse kernel parameters across all layers. We define the complete set of these learnable parameters as $\Theta = \{(\mathbf{o}_{l,i}, w_{l,i})\}_{l=1, i=1}^{L, N_l}$, which includes the offsets and weights for N_l samples in each of the L layers.

Our goal is to find the optimal parameters Θ^* by minimizing a loss function $\mathcal{L}$ that measures the discrepancy between our approximation and the target kernel:

$$\begin{aligned} \Theta^* &= \arg \min_{\Theta} \mathcal{L}(K_{target}, F_{approx}(\Theta)), \\ F_{approx}(\Theta) &= K_{s,1} * K_{s,2} * \dots * K_{s,L}, \end{aligned} \quad (4)$$

where K_{target} is the desired dense filter and $F_{approx}(\Theta)$ is the composite kernel formed by the convolution of the learned sparse kernels.

Learnable Parameter Our optimization strategy treats the offsets and weights of each sample as independent, learnable parameters. Specifically, for each layer l and for each of the N_l sampling points within it, we simultaneously optimize both the 2D offset vector $\mathbf{o}_{l,i}$ and its corresponding scalar weight $w_{l,i}$.

The complete set of learnable parameters for the entire model, denoted by Θ , is therefore the collection of all such offset-weight pairs:

$$\Theta = \bigcup_{l=1}^L \{(\mathbf{o}_{l,j}, w_{l,j})\}_{j=1}^{N_l}. \quad (5)$$

Initialization A well-designed parameter initialization is crucial for stable optimization convergence. Heuristic methods, such as Kawase (Kawase, 2003) and Dual Filtering (Martin et al., 2015), have fixed schemes tailored to specific filter types; however, a general approach is required for arbitrary target kernels of different sizes.

To address this, we propose a radial initialization strategy. The core idea is to initialize the sampling points in each layer to be uniformly distributed on the circumference of a circle, with the radius of this circle increasing linearly with the layer index. This progressive expansion ensures that the effective receptive field of the composite kernel grows with each subsequent layer, making the initial configuration capable of spanning a large-area target kernel from the outset. The radius for layer l , denoted r_l , is governed by a step size Δ_r derived from the target kernel’s spatial extent and the total number of layers L (see Appendix for derivation). The corresponding weights in each layer are initialized uniformly.

This initialization is formally defined as:

$$\begin{aligned}
r_l &= l \cdot \Delta_r && \text{for } l = 1, \dots, L, \\
\mathbf{o}_{l,i} &= \left(r_l \cos\left(\frac{2\pi i}{N_l}\right), r_l \sin\left(\frac{2\pi i}{N_l}\right) \right) && \text{for } i = 1, \dots, N_l, \\
w_{l,i} &= \frac{1}{N_l}.
\end{aligned} \tag{6}$$

4.2 SPARSE SAMPLING OF ARBITRARY KERNEL

A common way to initialize filter offsets is by sampling random positions within a local neighborhood. While this approach is general, it often traps the optimization in poor local optima, especially for kernels with complex or non-convex shapes.

Our method decomposes a dense kernel into a series of sparse ones. The first of these, $K_{s,1}$ (Eq. 4), has the greatest influence on the final filtered output, so its initialization is critical. A simple improvement over purely random sampling is to confine samples to the minimal bounding box of the kernel’s non-zero pixels. This ensures most samples fall near the target shape, but it is still inefficient for non-convex kernels, whose bounding boxes can contain large empty regions.

To overcome this limitation, we propose a more sophisticated initialization strategy leveraging rejection sampling. Instead of drawing samples from the kernel’s bounding box, our method samples directly from the support of the kernel, i.e., its non-zero locations. We first quantify the effective sampling area, denoted by S , as the count of these non-zero pixels. A sampling radius r is subsequently derived based on the desired number of samples, N_s :

$$r = \sqrt{\frac{S}{N_s \cdot \pi}}. \tag{7}$$

The detailed procedure is provided in the appendix. This initialization sets the first sparse kernel’s offsets to closely match the target shape. By restricting samples to relevant regions, it avoids vanishing gradients and reduces the risk of converging to poor local optima.

4.3 SPATIALLY VARYING FILTERING

Next, we propose a decomposition method for spatially varying filtering.

Spatially varying filtering generalizes convolution by applying a unique filter at each pixel (x, y) . The filter’s properties—such as its blur radius, orientation, or shape—are determined by a corresponding value $P(x, y)$ from a parameter map. The core challenge lies in efficiently synthesizing and applying these unique per-pixel kernels.

Conventional spatially varying filtering is often impractical. Generating dense kernels on the fly (Wang et al., 2023) is too slow, while precomputing them (Kovesi, 2010) incurs prohibitive memory costs; both are poorly suited to modern parallel hardware. Faster alternatives (Leimkühler et al., 2018) restrict filters to simple analytic forms (e.g., Gaussians), but this sacrifices the expressiveness needed for complex, non-convex point spread functions (PSFs).

We observe that, in prior work, the cost of generating or storing spatially varying kernels grows linearly with image resolution. To remove this bottleneck while retaining expressive, sparsely optimized kernels, we introduce *Filter-Space Interpolation*, which decouples kernel-generation complexity from image size.

Our spatially varying filtering is built on an ordered set of M basis sparse filters, which discretely sample a continuous, one-dimensional space $\mathcal{F}$ of filters. Each basis filter, f_k , corresponds to a scalar parameter p_k (with $p_1 < p_2 < \dots < p_M$) and consists of a unique set of N sampling offsets and weights. This design allows our basis to represent a wide range of filter behaviors across the parameter space, from applying arbitrary linear transformations to a kernel to simply varying the standard deviation (σ) of a Gaussian. We define the basis as:

$$\mathcal{F} = \{f_k(p_k) \mid k = 1, \dots, M\}, \quad \text{where} \quad f_k = \{(\mathbf{o}_{ki}, w_{ki})\}_{i=1}^N \tag{8}$$

We divide the approach into an offline pre-computation stage and a runtime inference stage. In the offline stage, we optimize each basis filter f_k individually to represent the ideal filter effect at its parameter value p_k .

At runtime, we synthesize a unique sparse filter for each pixel (x, y) , which is guided by a per-pixel parameter map, P . From the parameter value at each coordinate, $P(x, y)$, we determine a corresponding vector of M interpolation weights, $\alpha(x, y) = (\alpha_1, \dots, \alpha_M)$. These weights specify how to blend a compact set of basis filters, $\{f_k\}_{k=1}^M$, to reconstruct the final filter instance.

The final sparse filter for a given pixel, $f(x, y)$, is synthesized as a direct convex combination of the basis filters:

$$f(x, y) = \sum_{k=1}^M \alpha_k(x, y) \cdot f_k, \quad (9)$$

subject to the constraint that $\sum_{k=1}^M \alpha_k(x, y) = 1$ and $\alpha_k(x, y) \geq 0$.

By directly interpolating basis-filter offsets and weights, we sidestep the costly on-the-fly generation of kernels from analytical functions. This reduces the computational overhead of spatially varying kernel synthesis to a minimal set of parallelizable multiply-add operations. Furthermore, the interpolatable nature of our basis filters makes the entire set highly compressible, allowing us to significantly reduce the memory footprint required to achieve a wide range of expressive effects while offering flexible control over the quality-performance trade-off.

4.4 IMPLEMENTATION DETAILS

Training Process To ensure our learned filter parameters are generalized and not overfit to a specific dataset, we adopt an image-agnostic optimization strategy. We leverage a core principle of Linear Shift-Invariant (LSI) systems (Goodman, 2005): a filter is fully characterized by its impulse response.

First, we synthesize the effective kernel of our multi-pass filter, F_θ , by applying it to a discrete Dirac delta function, δ . The resulting output is the synthesized impulse response, K_{syn} . The impulse δ is an image with a single non-zero pixel at its center coordinate $\mathbf{c}$:

$$K_{\text{syn}} = F_\theta(\delta), \quad \text{where} \quad \delta[\mathbf{n}] = \begin{cases} 1 & \text{if } \mathbf{n} = \mathbf{c} \\ 0 & \text{otherwise.} \end{cases} \quad (10)$$

Here, θ represents the learnable parameters of our filter and $\mathbf{n}$ denotes the discrete pixel coordinates.

Loss Design Second, we define our loss function, $\mathcal{L}$, as the Charbonnier L1 loss $\mathcal{C}$ (Charbonnier et al., 1994) between the synthesized kernel K_{syn} and a target kernel K_{tgt} :

$$\mathcal{L} = \mathcal{C}(K_{\text{syn}}, K_{\text{tgt}}). \quad (11)$$

This impulse-response-based supervision allows us to "collapse" the entire multi-layer filtering sequence into a single, equivalent kernel for direct and precise approximation of the target.

5 EXPERIMENTS

In this section, we conduct a series of experiments to evaluate our differentiable kernel decomposition framework thoroughly. We first describe the experiment details and evaluation protocol in Section 5.1. Next, in Section 5.2, we assess our method's ability to approximate single, complex kernels, comparing it against state-of-the-art techniques. We extend this analysis to the more challenging task of spatially varying filtering in Section 5.3. To validate our specific design choices, we present a series of ablation studies in Section 5.4.

5.1 SETUP

Baselines. We compare our method against several baselines. For both single kernel and spatially varying filtering, we include a low-rank decomposition (LowRank) (McGraw, 2015) and the optimization-based method of Parallel Tempering (PST) (Schuster et al., 2020).

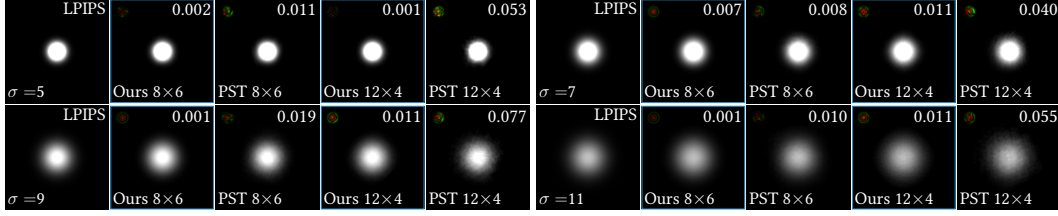


Figure 2: **Comparison of Gaussian kernel approximation with varying σ .** We compare our method against PST using two sparse configurations (8 layers $\times$ 6 samples and 12 layers $\times$ 4 samples). LPIPS scores appear in the top-right corner (lower is better). The top-left inset visualizes the error map, with positive errors shown in red and negative errors in green.

Datasets and Kernels. To evaluate the versatility of our method, we use a diverse set of target kernels and images. This set includes standard analytical shapes, such as Gaussian kernels (with σ values from 5 to 11). To assess performance on more complex targets, we additionally use a suite of arbitrary kernels comprising simple geometric primitives (disks, rings), regular polygons (4-sided and 6-sided), non-convex shapes (a heart, a four-pointed star, and an ampersand symbol), more complex shapes (animal silhouettes), and optical PSFs (coma and spherical aberration). For the spatially varying filtering experiments, we use five high-resolution photographs selected to represent realistic scenarios with complex textures and both 1D and 2D spatial variations.

Implementation and Evaluation Metric. We implement our methods in PyTorch and perform all optimization on a single GPU with 24 GB of memory, offering computational power comparable to an NVIDIA RTX 4090. For all configurations of kernels and layers, we use the same Adam optimizer with a learning rate linearly decayed from 1×10^{-3} to 1×10^{-4} . We use 1000 optimization steps per kernel for our method. For comparison, we run the PST algorithm for 10,000 iterations with 10 parallel candidates, for a total of 100,000 optimization steps. For the LowRank method, we utilize decompositions with ranks 1, 2 and 3, chosen to maintain a comparable number of samplings.

For runtime analysis, we benchmark our approach on a representative mobile device equipped with a Qualcomm Snapdragon 8 Gen 3 SoC, and report latency in milliseconds (ms). We evaluate both numerical fidelity and perceptual similarity using Peak Signal-to-Noise Ratio (PSNR), Learned Perceptual Image Patch Similarity (LPIPS) (Zhang et al., 2018), and FLIP-LDR (Andersson et al., 2020). Higher values indicate better performance for PSNR, and lower values are better for LPIPS and FLIP-LDR.

5.2 SINGLE KERNEL

Fig. 3 shows that our method consistently achieves a superior balance between reconstruction quality and inference speed compared to all other approaches. For our method and PST, the 'S', 'M', and 'L' correspond to total sample counts of 48 (12×4), 96 (24×4), and 128 (32×4), respectively. The LowRank's 'M' and 'L' use 98 (49×2) and 196 (49×4) parameters.

Next, we present a comparison of Gaussian kernel approximation with varying standard deviations σ in Fig. 2. In a 6-layer, 8-sample (8×6) configuration, our method achieves high-fidelity results with low perceptual error, whereas PST exhibits visible noise and artifacts. This performance gap widens in a sparser 12×4 setup. As σ increases, PST's approximation degrades severely, while our result remains visually coherent and maintains a substantially lower LPIPS error. These results show

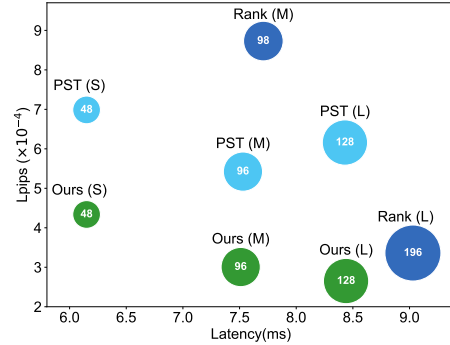


Figure 3: **Speed, accuracy, and samples comparison.** The figure plots quality against latency (lower is better for both). The size of each bubble represents the total sample count.

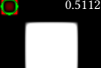
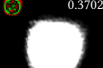
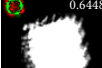
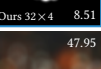
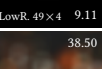
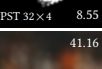
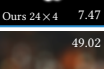
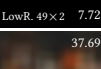
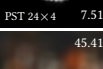
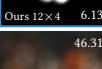
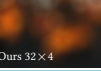
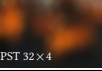
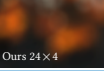
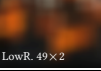
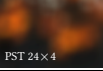
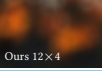
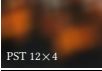
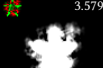
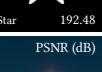
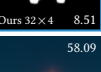
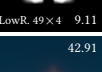
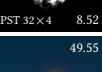
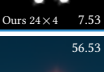
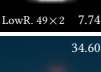
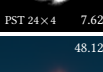
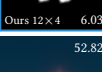
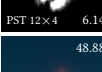
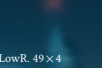
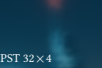
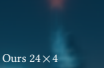
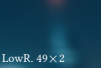
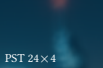
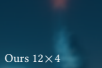
LPIPS ($\times 10^{-4}$)									
Disk	246.83	Ours 32 $\times$ 4 8.51	LowR. 49 $\times$ 4 9.11	PST 32 $\times$ 4 8.55	Ours 24 $\times$ 4 7.47	LowR. 49 $\times$ 2 7.72	PST 24 $\times$ 4 7.51	Ours 12 $\times$ 4 6.13	PST 12 $\times$ 4 6.13
PSNR (dB)		47.95	38.50	41.16	49.02	37.69	45.41	46.31	41.24
GT									
LPIPS ($\times 10^{-4}$)									
Star	192.48	Ours 32 $\times$ 4 8.51	LowR. 49 $\times$ 4 9.11	PST 32 $\times$ 4 8.52	Ours 24 $\times$ 4 7.53	LowR. 49 $\times$ 2 7.74	PST 24 $\times$ 4 7.62	Ours 12 $\times$ 4 6.03	PST 12 $\times$ 4 6.14
PSNR (dB)		58.09	42.91	49.55	56.53	34.60	48.12	52.82	48.88
GT									
LPIPS ($\times 10^{-4}$)									
Coma	243.58	Ours 32 $\times$ 4 8.36	LowR. 49 $\times$ 6 12.91	PST 32 $\times$ 4 8.68	Ours 24 $\times$ 4 7.45	LowR. 49 $\times$ 4 7.63	PST 24 $\times$ 4 7.57	Ours 12 $\times$ 4 6.15	PST 12 $\times$ 4 6.16
PSNR (dB)		49.09	39.61	36.51	47.10	39.61	39.61	48.21	36.99
GT									

Figure 4: **Comparison of Single kernel approximation.** Compared to baselines, SVD-based decomposition (LowR.) (McGraw, 2015) and Parallel Simulated Tempering (PST) (Schuster et al., 2020), our approach (blue) better preserves sharp features on non-convex targets, resulting in lower LPIPS scores (lower is better). The top-left inset visualizes the error map, with positive errors shown in red and negative errors in green.

that our gradient-based optimization yields more accurate approximations than PST, consistently producing stable solutions even in challenging sparse configurations.

Our method’s accuracy extends beyond Gaussian kernels to the more general case of arbitrary single-kernel filters, as shown in Fig. 4. Our method preserves structure across both simple and complex shapes, while LowRank introduces blocky artifacts and PST yields noisy results at low sample counts. Quantitatively, our method achieves the lowest LPIPS across all tests, often by a large margin. It is also much more efficient, requiring only 1/100 the iterations of PST.

5.3 SPATIALLY VARYING KERNEL

We present three spatially varying filtering examples in Fig. 5. The first is a 1D spatially varying blur that uses a pseudo-depth map to simulate a tilt-shift camera effect. The other two are 2D anisotropic effects: a rotational bokeh blur and a radial motion blur, both controlled by two parameters—blur intensity and local blur angle.

Our results are visually indistinguishable from the ground truth. As highlighted in the red and green insets, our method reproduces the complex structure of the GT kernels. In contrast, PST introduces noise and LowRank oversmooths, and neither recovers the correct kernel shape, while directly applying GT kernels is prohibitively slow. Quantitatively, our method achieves the highest PSNR among all methods while maintaining real-time performance.

This performance difference stems from how well each method’s base kernels handle filter-space interpolation. While all approaches use interpolation to generate the varying filter parameters, our optimization-based kernels are better conditioned for this process and appear to vary more linearly. Consequently, they interpolate smoothly to form sharp, complex patterns. PST’s kernels, however, suffer from poor optimization quality, and interpolating between them simply produces more noise. Similarly, interpolating the basis kernels from LowRank’s decomposition causes them to average into indistinct blurs rather than preserving the target structure.

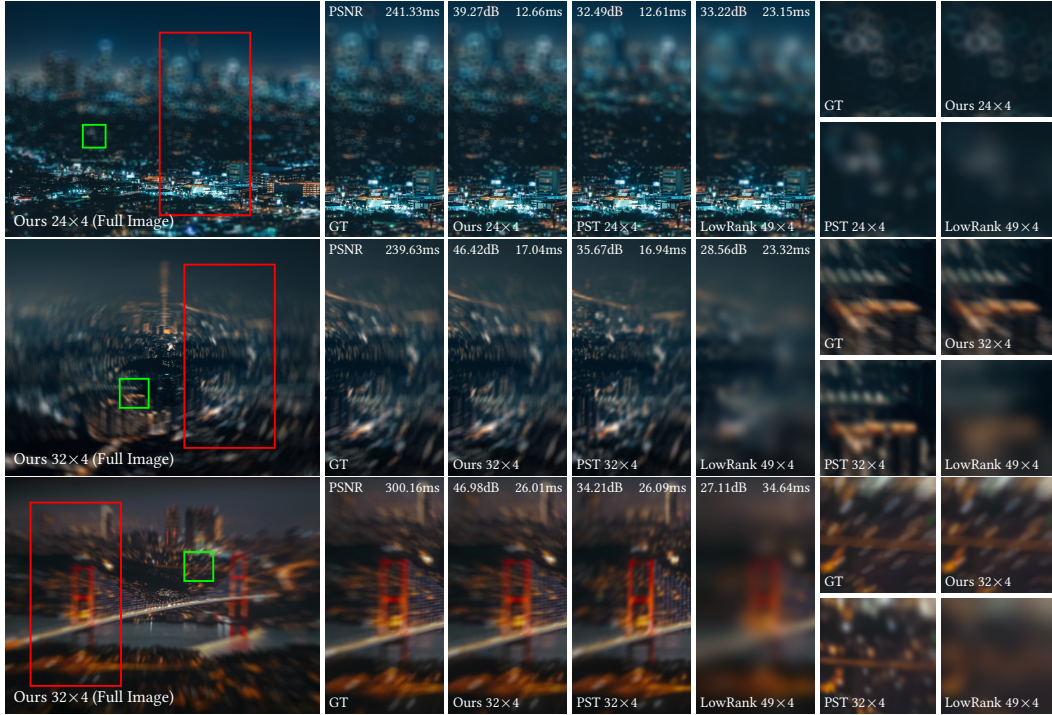


Figure 5: **Visual comparison of diverse spatially varying (SV) effects.** We evaluate three SV configurations: 1D tilt-shift blur (top), 2D rotational blur (middle), and 2D radial motion blur (bottom). We compare our method against Parallel Simulated Tempering (PST) and Low-Rank Decomposition (LowRank).

5.4 ABLATIONS

We conduct ablation studies to validate our main design choices, focusing on both initialization strategies and different layer configurations.

We first evaluate different initialization schemes across multiple kernels, as shown in Fig. 6. Both our method and Parallel Simulated Tempering (PST) benefit from the proposed Sparse Sampling (SS) initialization, which consistently outperforms the Increasing Radial (IR) initialization, while the Random (Rand) initialization performs worst. Although SS accelerates convergence for both our method and PST, PST still requires more than 30× the number of iterations to converge compared with ours, and our final reconstruction quality is significantly higher.

We further study the influence of different configurations, varying the number of layers and the number of samples, as shown in Fig. 7. The convergence curves show that all configurations converge stably, and configurations with more samples and layers tend to achieve higher quality. Compared with PST, our method delivers more consistent behavior and better quality across all tested configurations.

For additional results, please refer to the Appendix, which includes ablations on Gaussian kernels with fewer samples and quantitative evaluations of initialization and regularization strategies on arbitrary kernels.

6 DISCUSSION AND CONCLUSION

We introduced a differentiable framework that recasts the challenging problem of approximating large, complex convolution kernels as an end-to-end optimization task. Our approach supports a wide range of kernels—from simple Gaussians to complex, non-convex forms—and converges to high-fidelity solutions far more efficiently than prior methods. We extend this with filter-space in-

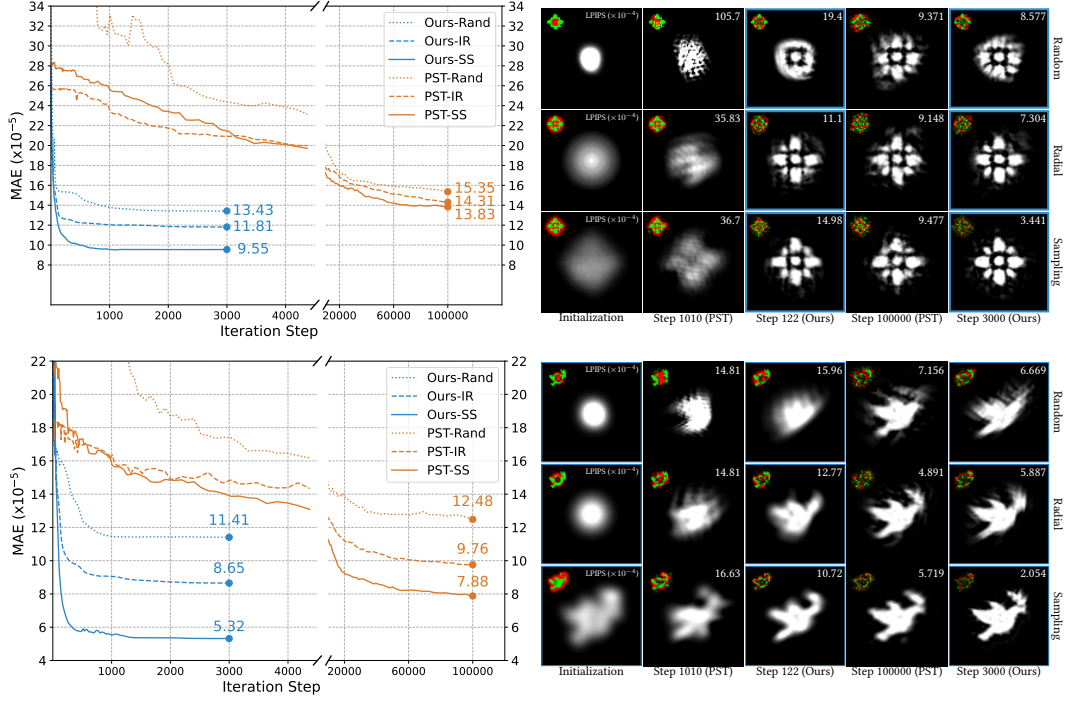


Figure 6: **Ablation of initialization strategies on the *Flower* and *Dove* kernel.** We evaluate both our method and Parallel Simulated Annealing (PST) combined with three initialization schemes: Random (Rand), Increasing Radial (IR), and Sparse Sampling (SS).

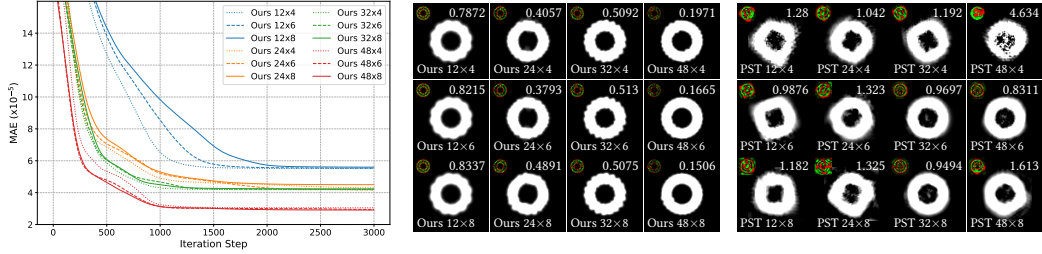


Figure 7: **Ablation results for various configurations of samples and layers on Ring kernel.**

terpolation, enabling multi-dimensional spatially varying effects with minimal per-pixel overhead. A current constraint of our formulation is that it requires access to the target dense kernel during optimization and filtering. This work opens several promising avenues for future research, including multi-dimensional parameter maps for simultaneous control over kernel attributes and the use of neural architecture search to discover hardware-optimized filter decompositions. Overall, our method provides a practical, high-performance solution for advanced image filtering in real-time applications such as computational photography, while remaining fully differentiable and thus usable as a trainable layer within modern deep learning pipelines.

ACKNOWLEDGMENTS

This work was partially supported by National Key R&D Program of China (No. 2024YFB2809104), NSFC (No. 52532013), and Key R&D Program of Zhejiang (No. 2025C01064). We also thank the anonymous reviewers for their constructive comments.

REFERENCES

- Pontus Andersson, Jim Nilsson, Tomas Akenine-Möller, Magnus Oskarsson, Kalle Åström, and Mark D Fairchild. Flip: A difference evaluator for alternating images. *Proc. ACM Comput. Graph. Interact. Tech.*, 3(2):15–1, 2020.
- Pierre Charbonnier, Laure Blanc-Feraud, Gilles Aubert, and Michel Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. In *Proceedings of 1st international conference on image processing*, volume 2, pp. 168–172. IEEE, 1994.
- François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.
- Joseph W Goodman. *Introduction to Fourier optics*. Roberts and Company publishers, 2005.
- Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
- Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. *Advances in neural information processing systems*, 29, 2016.
- Masaki Kawase. Frame buffer postprocessing effects in double-steal (wrechless). In *Game Developers Conference 2003*, 3, 2003.
- Peter Kovesi. Fast almost-gaussian filtering. In *2010 International conference on Digital image computing: Techniques and applications*, pp. 121–125. IEEE, 2010.
- Thomas Leimkühler, Hans-Peter Seidel, and Tobias Ritschel. Laplacian kernel splatting for efficient depth-of-field and motion blur synthesis or reconstruction. *ACM Transactions on Graphics (TOG)*, 37(4):1–11, 2018.
- Yang Li, Bobo Yan, Jianxin Hou, Bingyang Bai, Xiaoyu Huang, Canfei Xu, and Limei Fang. Unet based on dynamic convolution decomposition and triplet attention. *Scientific Reports*, 14(1):271, 2024.
- Yunsheng Li, Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Ye Yu, Lu Yuan, Zicheng Liu, Mei Chen, and Nuno Vasconcelos. Revisiting dynamic convolution via matrix decomposition. *arXiv preprint arXiv:2103.08756*, 2021.
- Cewen Liu, Mengyao Sun, Nanxun Dai, Wei Wu, Yanwen Wei, Mingjie Guo, and Haohuan Fu. Deep learning-based point-spread function deconvolution for migration image deblurring. *Geophysics*, 87(4):S249–S265, 2022.
- Pinjun Luo, Guoqiang Xiao, Xinbo Gao, and Song Wu. Lkd-net: Large kernel convolution network for single image dehazing. In *2023 IEEE international conference on multimedia and expo (ICME)*, pp. 1601–1606. IEEE, 2023.
- Sam Martin, Andrew Garrard, Andrew Gruber, Marius Bjorge, Renaldas Zioma, Simon Benge, and Niklas Nummelin. Moving mobile graphics. In *ACM SIGGRAPH 2015 Courses*, SIGGRAPH ’15, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336345. doi: 10.1145/2776880.2787664. URL <https://doi.org/10.1145/2776880.2787664>.
- Tim McGraw. Fast bokeh effects using low-rank linear filters. *The Visual Computer*, 31(5):601–611, 2015.
- Ben Mildenhall, Jonathan T Barron, Jiawen Chen, Dillon Sharlet, Ren Ng, and Robert Carroll. Burst denoising with kernel prediction networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2502–2510, 2018.
- Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive convolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 670–679, 2017.

-
- Kurniawan Nur Ramadhani, Rinaldi Munir, and Nugraha Priya Utama. Improving video vision transformer for deepfake video detection using facial landmark, depthwise separable convolution and self attention. *IEEE Access*, 12:8932–8939, 2024.
- Parikshit Vishwas Sakurikar. Epsilon focus photography a study of focus defocus and depth of field.
- Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty R Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacher, Aaron Lefohn, and Marco Salvi. Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination. In *Proceedings of High Performance Graphics*, pp. 1–12. 2017.
- Kersten Schuster, Philip Trettner, and Leif Kobbelt. High-performance image filters via sparse approximations. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 3(2):1–19, 2020.
- Adrian Shajkofci and Michael Liebling. Spatially-variant cnn-based point spread function estimation for blind deconvolution and depth estimation in optical microscopy. *IEEE Transactions on Image Processing*, 29:5848–5861, 2020.
- Chao Wang, Krzysztof Wolski, Xingang Pan, Thomas Leimkühler, Bin Chen, Christian Theobalt, Karol Myszkowski, Hans-Peter Seidel, and Ana Serrano. An implicit neural representation for the image stack: Depth, all in focus, and high dynamic range. Technical report, 2023.
- Yi Wu, Yanyang Xu, Wenhao Zhu, Guojie Song, Zhouchen Lin, Liang Wang, and Shaoguo Liu. Kdlt: A linear graph transformer framework via kernel decomposition approach. In *IJCAI*, pp. 2370–2378, 2023.
- Zijin Wu, Xingyi Li, Juewen Peng, Hao Lu, Zhiguo Cao, and Weicai Zhong. Dof-nerf: Depth-of-field meets neural radiance fields. In *Proceedings of the 30th ACM International Conference on Multimedia*, pp. 1718–1729, 2022.
- Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.
- Jingkai Zhou, Varun Jampani, Zhixiong Pi, Qiong Liu, and Ming-Hsuan Yang. Decoupled dynamic filter networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6647–6656, 2021.
- Shangchen Zhou, Jiawei Zhang, Jinshan Pan, Haozhe Xie, Wangmeng Zuo, and Jimmy Ren. Spatiotemporal filter adaptive network for video deblurring. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 2482–2491, 2019.
- A Zing. Extended binomial filter for fast gaussian blur. *Vienna, Austria*, 2010.

A SAMPLING ALGORITHM OF KERNEL SHAPE

Algorithm 1 Non-Convex Kernel Initialization via Rejection Sampling

```

1: Input: Target kernel  $K$ , Number of samples  $N_s$ 
2: Output: Initial offsets for the first sparse kernel  $O_0$ 
3:  $M \leftarrow$  Coordinates of non-zero pixels in  $K$ 
4:  $S \leftarrow |M|$ 
5:  $r \leftarrow \sqrt{S/(N_s \cdot \pi)}$  {Initial rejection radius}
6:  $O_0 \leftarrow \emptyset$  {Initialize empty set for offsets}
7: for each candidate coordinate  $p \in M$  do
8:   if  $|O_0| \geq N_s$  then
9:     break
10:  end if
11:  if  $\forall o \in O_0, \|p - o\| \geq r$  then
12:     $O_0 \leftarrow O_0 \cup \{p\}$  {Accept sample if it's not too close to others}
13:  end if
14: end for
15: if  $|O_0| < N_s$  then
16:    $N_{\text{rem}} \leftarrow N_s - |O_0|$  {Handle case where not enough samples were found}
17:    $M_{\text{fill}} \leftarrow$  First  $N_{\text{rem}}$  coordinates from a deterministic repetition of  $M$ 
18:    $O_0 \leftarrow O_0 \cup M_{\text{fill}}$ 
19: end if
20:  $O_0 \leftarrow O_0 - \text{center}(K)$  {Center the offsets}
21: return  $O_0$ 

```

B ADDITIONAL RESULT

B.1 ABLATION ON MULTIPLE CONFIGS OF SAMPLES AND LAYERS

We present ablation of samples and layers on another kernel, *Dove*, in Fig. 8.

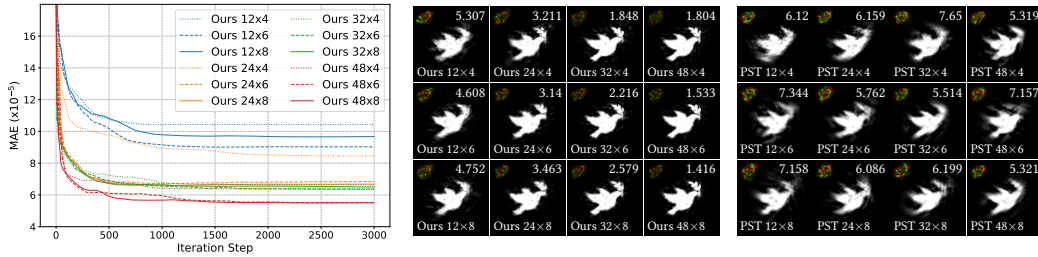


Figure 8: Ablation results for various configurations of samples and layers on *Dove* kernel.

B.2 ABLATION ON GAUSSIAN KERNEL OF FEWER SAMPLES

For symmetric targets like Gaussians, we enforce a Kawase-like (KWS) symmetry constraint on the sampling offsets. Table 1 shows this constraint consistently improves results, proving particularly effective when using only a few samples per layer. We also found that an L1 loss on the impulse response is substantially more effective than an L2 loss for this task. Our final model for Gaussian kernels with fewer samples (e.g., 6-layer and 4 samples per layer) can combine an L1 loss with symmetry constraint and IR strategies.

B.3 INITIALIZATION ABLATION ON VARIOUS KERNELS

As shown in Table 2, a basic random initialization performs poorly. Using either our sparse sampling (SS) for the first layer or our increasing-step radial initialization (IR) for all layers provides a

Table 1: Ablation results of filtering of Gaussian kernels on impulse response.

Variant \ Filter	$\sigma = 5$			$\sigma = 7$			$\sigma = 9$			$\sigma = 11$		
	PSNR $\uparrow$	LPIPS $\downarrow$	FLIP $\downarrow$	PSNR $\uparrow$	LPIPS $\downarrow$	FLIP $\downarrow$	PSNR $\uparrow$	LPIPS $\downarrow$	FLIP $\downarrow$	PSNR $\uparrow$	LPIPS $\downarrow$	FLIP $\downarrow$
L1+KWS+IR	97.79	.0099	.2604	99.98	.0052	.4943	99.85	.0028	.1906	96.77	.0063	.3926
L1+KWS+R	96.11	.0150	.5567	93.46	.0181	.6603	90.57	.0256	.4152	79.72	.2737	2.670
L1+IR	98.31	.0117	.3871	93.69	.0394	.7209	91.83	.1142	.9803	91.99	.0942	.7191
L1+R	97.64	.0191	.4443	93.05	.0705	.7021	90.77	.1606	.9916	88.16	.3198	1.226
L2+KWS+IR	65.04	19.57	6.635	67.77	1.393	7.119	69.96	.3123	7.156	71.80	.1318	6.576
L2+Inc	90.18	.0849	.8460	90.02	.1433	1.066	83.84	.2396	2.291	77.17	.4472	4.389

significant boost, but combining them as SS+IR is critical for achieving the best performance. This confirms our hypothesis that a hybrid strategy is optimal, with SS capturing kernel shape and IR ensuring stable receptive field growth. Furthermore, weight normalization is crucial; both sum-to-one (SUM) and softmax (SOFM) are effective, with SUM holding a slight, consistent advantage.

Table 2: **Ablation on arbitrary kernels.** Our full model (SUM+SS+IR) achieves the best performance. SUM/SOFM: sum-to-one/softmax weight normalization. SS: sparse sampling initialization. IR/R: increasing/fixed-step radial initialization. Basic: random initialization. For LPIPS and FLIP scores, both lower is better.

Variant \ Filter	Ampersand		Disk		Heart		Star4		Ring		4-Sided		6-Sided		Star	
	LPIPS	FLIP	LPIPS	FLIP	LPIPS	FLIP	LPIPS	FLIP	LPIPS	FLIP	LPIPS	FLIP	LPIPS	FLIP	LPIPS	FLIP
SUM+SS+IR	1.44	2.63	.240	3.53	.230	1.54	.326	3.00	.426	2.46	.276	.375	.218	1.77	.790	5.28
SOFM+SS+IR	2.39	2.61	.242	3.54	.248	1.58	.311	3.07	.432	2.72	.299	.420	.246	2.04	.798	5.38
SUM+SS+R	1.94	3.12	.251	3.48	.236	1.50	.386	3.48	.441	2.60	.285	.492	.336	2.10	.748	5.60
SOFM+SS+R	1.90	2.97	.269	3.81	.241	1.41	.466	3.38	.434	2.48	.332	.481	.374	2.67	.937	5.93
SUM+R	9.74	11.7	.774	14.6	1.13	8.78	1.52	8.88	2.57	25.0	1.65	5.06	.960	10.4	6.44	11.1
SOFM+R	9.70	11.7	.773	14.6	1.13	8.78	1.52	8.88	2.57	25.0	1.65	5.06	.961	10.4	6.44	11.0
SS	10.4	10.3	.524	22.6	.906	11.2	2.16	15.1	2.28	10.3	1.20	12.8	.744	12.6	4.62	7.61
Basic	21.1	25.3	12.0	32.8	13.9	26.7	23.2	17.3	233	38.4	11.3	23.1	9.83	29.1	13.8	18.8

B.4 SINGLE KERNEL COMPARISON

We present additional results on single-kernel synthesis in Fig. 9. We include a diverse set of point-spread functions, ranging from basic depth-of-field bokeh shapes (*Side4*, *Side6*, *Ring*) to more complex geometric silhouettes (*Heart*, *Star4*), an artificial font-based shape (*Ampersand*), and finally dove- and flower-shaped apertures and an optical PSF with spherical aberration. This progression illustrates that our sparse kernels can handle both simple analytic shapes and highly non-Gaussian, visually intricate kernels. We also compare three configurations of our method (32×4 , 24×4 , and 12×4); while 32×4 generally provides the best reconstruction quality, the 12×4 variant still recovers the main structures despite using very few samples, indicating that our representation retains reasonable expressive power even under an extremely small sample budget.

B.5 COMPARISON ON SPATIAL VARYING FILTERING

We quantitatively evaluate our filter-space interpolation in Fig. 10, plotting LPIPS error against a continuous kernel size parameter. For all tested kernel shapes, our method achieves lower error than both baselines. The LowRank method is not well-suited for this continuous approximation task, as it exhibits high error across all sizes. The PST baseline is more competitive but remains consistently less accurate than our approach, particularly as the kernel size increases.

B.6 MEMORY CONSUMPTION ANALYSIS

Side4	LPIPS ($\times 10^{-4}$)		0.2776		1.501		0.9742		0.1593		1.939		0.6902		0.004485		0.4794	
	GT	241.39	Ours 32 $\times$ 4	8.48	LowR 49 $\times$ 4	9.01	PST 32 $\times$ 4	8.28	Ours 24 $\times$ 4	7.51	LowR 49 $\times$ 2	7.69	PST 24 $\times$ 4	7.59	Ours 12 $\times$ 4	6.26	PST 12 $\times$ 4	6.23
Side6	LPIPS ($\times 10^{-4}$)		0.2369		0.3574		0.4807		0.398		0.6766		0.4697		0.4707		0.6378	
	GT	300.83	Ours 32 $\times$ 4	8.47	LowR 49 $\times$ 4	9.02	PST 32 $\times$ 4	8.41	Ours 24 $\times$ 4	7.58	LowR 49 $\times$ 2	7.74	PST 24 $\times$ 4	7.57	Ours 12 $\times$ 4	6.13	PST 12 $\times$ 4	6.07
Ring	LPIPS ($\times 10^{-4}$)		0.453		1.085		1.193		0.4057		2.549		1.128		0.7872		1.079	
	GT	241.38	Ours 32 $\times$ 4	8.43	LowR 49 $\times$ 4	8.93	PST 32 $\times$ 4	8.41	Ours 24 $\times$ 4	7.48	LowR 49 $\times$ 2	7.74	PST 24 $\times$ 4	7.42	Ours 12 $\times$ 4	6.09	PST 12 $\times$ 4	6.11
Heart	LPIPS ($\times 10^{-4}$)		0.3858		0.857		0.6456		0.3585		1.107		0.5122		0.4867		0.8572	
	GT	241.36	Ours 32 $\times$ 4	8.52	LowR 49 $\times$ 4	9.1	PST 32 $\times$ 4	8.48	Ours 24 $\times$ 4	7.45	LowR 49 $\times$ 2	7.57	PST 24 $\times$ 4	7.45	Ours 12 $\times$ 4	6.12	PST 12 $\times$ 4	6.13
Star4	LPIPS ($\times 10^{-4}$)		0.3425		0.7456		0.8514		0.4191		1.94		0.8856		0.6864		0.6579	
	GT	241.4	Ours 32 $\times$ 4	8.29	LowR 49 $\times$ 4	8.9	PST 32 $\times$ 4	8.35	Ours 24 $\times$ 4	7.58	LowR 49 $\times$ 2	7.77	PST 24 $\times$ 4	7.59	Ours 12 $\times$ 4	6.24	PST 12 $\times$ 4	6.07
Amperсанд	LPIPS ($\times 10^{-4}$)		1.86		5.852		3.583		2.691		8.848		5.826		4.438		4.548	
	GT	241.43	Ours 32 $\times$ 4	8.48	LowR 49 $\times$ 4	9.04	PST 32 $\times$ 4	8.54	Ours 24 $\times$ 4	7.47	LowR 49 $\times$ 2	7.74	PST 24 $\times$ 4	7.41	Ours 12 $\times$ 4	6.21	PST 12 $\times$ 4	6.24
Flower	LPIPS ($\times 10^{-4}$)		3.451		5.316		9.371		5.495		8.79		9.727		13.17		10.63	
	GT	248.85	Ours 32 $\times$ 4	8.61	LowR 49 $\times$ 6	12.91	PST 32 $\times$ 4	8.61	Ours 24 $\times$ 4	7.53	LowR 49 $\times$ 4	7.63	PST 24 $\times$ 4	7.56	Ours 12 $\times$ 4	6.13	PST 12 $\times$ 4	6.02
Dove	LPIPS ($\times 10^{-4}$)		1.837		8.815		7.65		3.061		13.03		6.159		5.306		6.12	
	GT	241.08	Ours 32 $\times$ 4	8.32	LowR 49 $\times$ 6	12.67	PST 32 $\times$ 4	8.56	Ours 24 $\times$ 4	7.57	LowR 49 $\times$ 4	7.77	PST 24 $\times$ 4	7.55	Ours 12 $\times$ 4	6.11	PST 12 $\times$ 4	6.05
Spherical	LPIPS ($\times 10^{-4}$)		4.789		11.95		25.29		6.737		48.77		58.73		15.17		28.84	
	GT	247.31	Ours 32 $\times$ 4	8.39	LowR 49 $\times$ 6	13.01	PST 32 $\times$ 4	8.56	Ours 24 $\times$ 4	7.5	LowR 49 $\times$ 4	7.75	PST 24 $\times$ 4	7.59	Ours 12 $\times$ 4	6.21	PST 12 $\times$ 4	6.11

Figure 9: Additional comparison of the single, arbitrary kernels.

We report the memory read and write bandwidth, along with their relative overhead compared to directly displaying the input image, in Table 3. All measurements are collected on a Snapdragon 8 Gen 3 mobile platform at a resolution of 1264×2665 .

Table 3: **Memory bandwidth and runtime comparison.** RB and WB denote read and write bandwidth (MB/frame), while RR and WR represent the *relative increase* in bandwidth with respect to the *Identical* (point sampling) baseline, rather than the raw ratio. Time indicates per-frame runtime (ms). The superscript * marks 1D spatially varying (SV) kernels. $S \times P$ implies S samples across P passes.

	Identical	GT*	24 $\times$ 4*	32 $\times$ 4*	49 $\times$ 4*	GT	12 $\times$ 4	24 $\times$ 4	32 $\times$ 4	49 $\times$ 2	49 $\times$ 4	49 $\times$ 6
RB	31.68	67.75	53.47	52.92	49.99	66.83	44.31	47.97	43.95	37.90	42.11	48.52
WB	25.63	90.27	41.56	41.56	39.92	89.17	29.11	32.52	28.20	30.62	33.10	35.16
RR(%)	0	113.8	68.78	67.05	57.80	110.9	72.88	87.16	71.48	47.87	64.30	89.31
WR(%)	0	252.2	62.15	62.15	55.75	247.9	13.58	26.88	10.03	19.47	29.15	37.18
Time	1.04	192.4	12.66	16.78	23.29	246.8	6.13	7.50	8.36	7.72	9.11	12.56

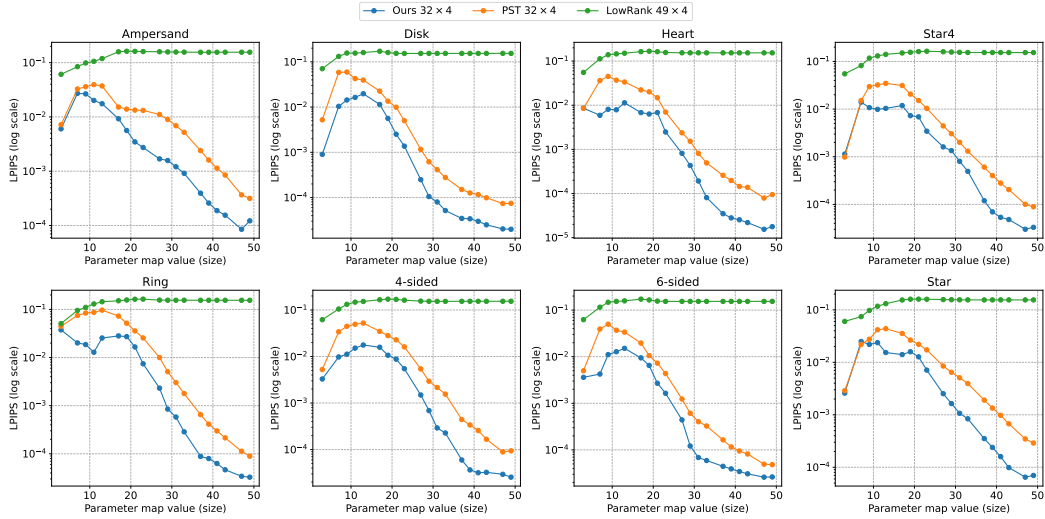


Figure 10: **Quantitative evaluation of spatially-varying kernel approximation.** We plot LPIPS error (Y-axis, log scale, lower is better) against a continuous kernel size parameter (X-axis) for eight different kernel shapes.

B.7 COMPARISON TO DILATED CONVOLUTIONS AND U-NETS

Fig. 11 compares our sparse kernel representation to dilated convolutions and U-Nets for single-kernel synthesis. For the CNN baselines, we reduce the learning rate to one-tenth of ours and increase the number of iterations from 3k to 10k to improve stability. Overall, our sparse kernels achieve comparable or lower LPIPS errors with substantially fewer parameters, suggesting that flexible sampling locations are important for representing such complex PSFs.

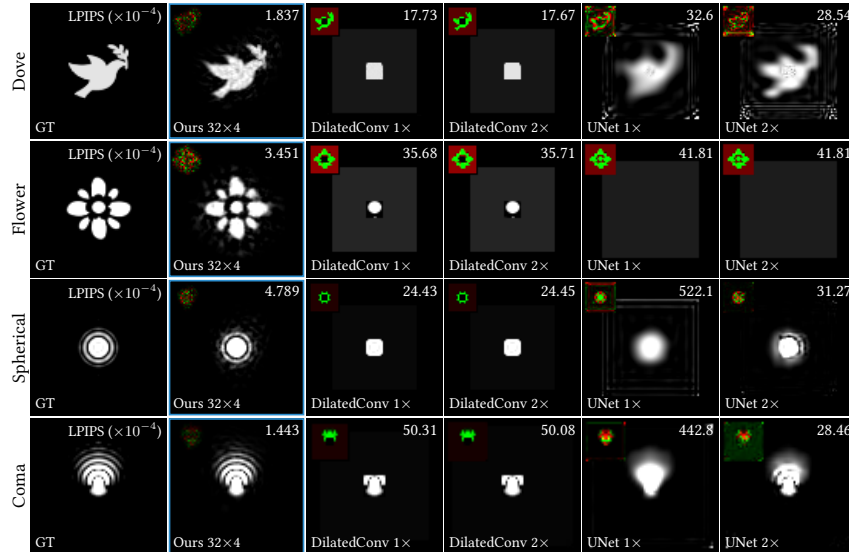


Figure 11: **Comparison of sparse kernels to CNN baselines on single-kernel synthesis.** For four representative non-Gaussian PSFs (Dove, Flower, Spherical, and Coma), we show the ground-truth kernel and the reconstructions produced by our sparse kernel (Ours 32 $\times$ 4), two dilated-convolution networks, and two U-Nets. Numbers in the top-left of each panel denote LPIPS error ($\times 10^{-4}$, lower is better). The number of trainable parameters is 384 for Ours 32 $\times$ 4, 75,073 for DilatedConv 1 $\times$, 297,601 for DilatedConv 2 $\times$, 7,705 for U-Net 1 $\times$, and 17,254 for U-Net 2 $\times$.