Learning Steerable Function for Efficient Image Resampling

Jiacheng Li[1*] Chang Chen[2*] Wei Huang[1]
Zhiqiang Lang[2] Fenglong Song[2] Youliang Yan[2] Zhiwei Xiong[1†]
[1]University of Science and Technology of China [2]Huawei Noah's Ark Lab
{jclee,weih527}@mail.ustc.edu.cn zwxiong@ustc.edu.cn
{chenchang25,langzhiqiang,songfenglong,yanyouliang}@huawei.com

Abstract

Image resampling is a basic technique that is widely employed in daily applications. Existing deep neural networks (DNNs) have made impressive progress in resampling performance. Yet these methods are still not the perfect substitute for interpolation, due to the issues of efficiency and continuous resampling. In this work, we propose a novel method of Learning Resampling Function (termed LeRF), which takes advantage of both the structural priors learned by DNNs and the locally continuous assumption of interpolation methods. Specifically, LeRF assigns spatially-varying steerable resampling functions to input image pixels and learns to predict the hyper-parameters that determine the orientations of these resampling functions with a neural network. To achieve highly efficient inference, we adopt look-up tables (LUTs) to accelerate the inference of the learned neural network. Furthermore, we design a directional ensemble strategy and edge-sensitive indexing patterns to better capture local structures. Extensive experiments show that our method runs as fast as interpolation, generalizes well to arbitrary transformations, and outperforms interpolation significantly, e.g., up to 3dB PSNR gain over bicubic for $\times 2$ upsampling on Manga109.

1. Introduction

Due to the rapid growth of visual data, there is a strong demand for digital image processing. Image resampling, one of the most common techniques, aims to obtain another image by generating new pixels following a geometric transformation rule from existing pixels in a given image [8]. Common transformations include upsampling (*i.e.*, single image super-resolution), downsampling, affine transformation, etc. Image resampling enjoys various applications, ranging from photo editing, optical distortion compensation [10], online content streaming [35], and visual special effects production [38].

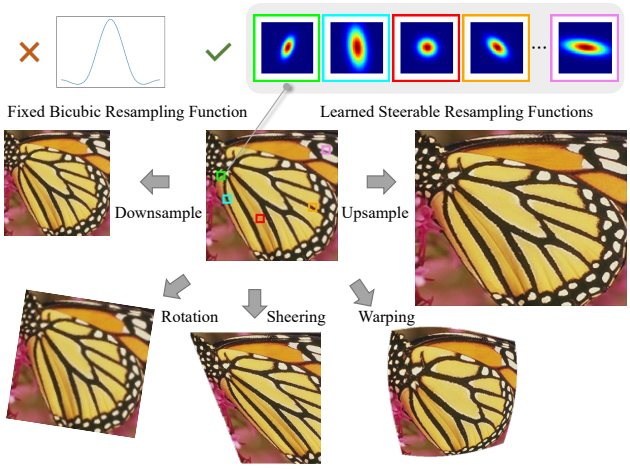


Figure 1. LeRF assigns steerable resampling functions to input pixels, and learns to predict the hyper-parameters that determine the orientations of these continuous functions for resampling under arbitrary transformations.

Recently, deep neural networks (DNNs) have made impressive progress in the field of image resampling [9, 12, 17, 27, 39, 40], thanks to the learning-from-data paradigm that obtains powerful structural priors from large-scale datasets. Despite the superior performance that DNN-based methods have achieved, long-lived interpolation methods like bicubic [16] are still preferred choices in most cases.

We attribute this phenomenon to the following two reasons: 1) Interpolation is simple and highly efficient, resulting in less dependency and thus the practicality to be deployed on a variety of devices, ranging from IoT devices to gaming workstations. 2) Interpolation supports arbitrary transformations. It assumes a continuous resampling function for a local area, resulting in the versatility in applying to not only homographic transformations like upsampling and downsampling, but also general warping. Although recent DNN-based methods explore beyond fixed-scale upsampling [5, 12, 27, 39, 43, 48], an efficient and continuous

*Equal contribution. †Corresponding author. This work was done when Jiacheng Li was a research intern at Huawei Noah's Ark Lab.

solution that matches interpolation remains less explored.

In this work, we aim to fill this blank research area by taking a middle way between DNN-based methods and interpolation methods. We propose a novel method of Learning Resampling Function (termed LeRF), where parameterized continuous functions for resampling different structures are learned from data. Specifically, as illustrated in Fig. 1, we assign spatially-varying steerable resampling functions to image pixels, whose orientations are parameterized with several hyper-parameters. Then, we train a neural network to predict these hyper-parameters for each pixel in an input image, thus defining the resampling function for that pixel location. Finally, we obtain the output image by interpolating an input image with these locally adapted resampling functions. LeRF takes advantage of both the structural priors learned by DNN and the locally continuous assumption of interpolation methods. Furthermore, we present an efficient implementation, where the inference of the learned neural network is accelerated with look-up tables (LUTs) [15, 22, 23, 29]. We further design a directional ensemble strategy and edge-sensitive indexing patterns to better capture local structures in images.

We examine the advantages and generalization ability of LeRF in various image resampling tasks, including arbitrary-scale upsampling, homographic transformation, and general warping. In particular, as illustrated in Fig. 2, at a similar running time, our method outperforms popular interpolation methods significantly in upsampling, which demonstrates the superiority of LeRF in terms of performance and efficiency.

Contributions of this paper are summarized as follows:

1) We propose LeRF, a novel method for continuous resampling. We assign spatially-varying steerable resampling functions to image pixels, where we train a neural network to predict the hyper-parameters that determine the orientations of these resampling functions.

2) We present an efficient implementation of LeRF by adopting look-up tables to accelerate the inference of the trained neural network. Furthermore, we design a directional ensemble strategy and edge-sensitive indexing patterns to better capture local structures.

3) Extensive experiments demonstrate that our method operates as efficient as interpolation, generalizes well to arbitrary transformations, and obtains significantly better performance over interpolation.

2. Related Works

Image interpolation for resampling. Interpolation, the most common solution for image resampling, assumes a locally continuous intensity surface and approximates it with a fixed resampling function, such as `nearest` (nearest), `linear` (bilinear), `cubic` (bicubic) [16], and windowed `sinc` (lanczos). It predicts resampling weights with the assumed resampling function, and then aggregates input pixels with these resampling weights to obtain the target pixel. This assumption allows for continuous resampling under arbitrary transformations, yet leads to blurry results due to ignoring different local structures [34]. We follow the same assumption on local continuity, but our method deals with different structures with adapted resampling functions instead of fixed ones.

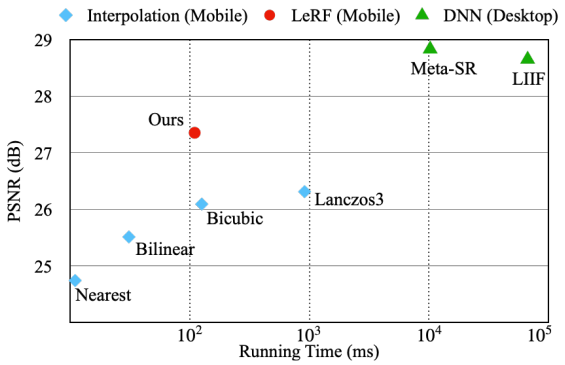
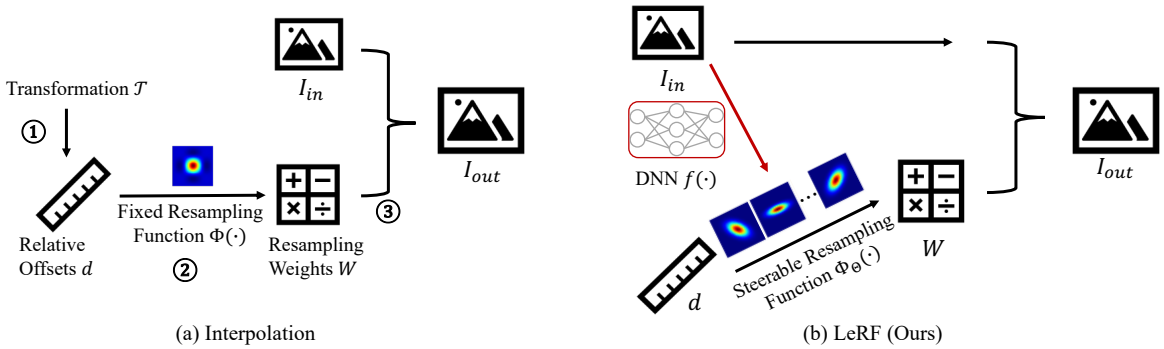


Figure 2. Performance-efficiency trade-off of arbitrary-scale upsampling methods. PSNR values are obtained on Set14 for $\times 4$ upsampling. The running time is evaluated on mobile and desktop devices, respectively.

Adaptive image resampling. To integrate local structural information into the resampling process, many methods are proposed, including edge-directed interpolation [2, 24, 44, 52] and kernel regression [21, 41, 51]. Different from these methods that rely on hand-designed rules, our method adopts a neural network to learn structural priors and integrate them into the resampling functions in a data-driven way. Another line of work achieves adaptive resampling by combining interpolation methods with adaptive filtering [11, 14, 17, 37]. Among them, RAISR achieves super-resolution by predicting adaptive filters for each pixel from a hash table and then applying the predicted adaptive filters to pre-upsampled images [11, 37]. This kind of method operates on the pre-resampled results of common interpolation methods (usually bicubic) with a fixed transformation (*e.g.*, $\times 2$ upsampling), thus lacking the generalization ability to be extended to unseen arbitrary transformations.

DNN-based image resampling. With the rise of deep neural networks, impressive progress has been made in image resampling, such as upsampling [3, 4, 9, 17, 25, 27, 36, 46, 47, 49, 54], downsampling [40, 42], and homographic transformation [39]. For arbitrary-scale upsampling, Meta-SR [12] and LIIF [5] are two representative works. Meta-SR predicts the resampling weights from upsampling scales with a weight prediction network, similar to kernel prediction networks [26, 32], and aggregates deep features to get the final upsampled image [12]. Following works improve the process with support for homographic transformations [39] and scale-aware feature adaptation [43]. On the other hand, LIIF models the aggregation step with a neural implicit rep-



(a) Interpolation (b) LeRF (Ours)

Figure 3. Comparison between interpolation and LeRF. (a) Interpolation assumes a spatial-invariant fixed function $\Phi(\cdot)$ to predict resampling weights W. (b) LeRF learns spatially-varying steerable function Φ_Θ with a DNN $f(\cdot)$.

resentation, predicting the target pixels from deep latent features and target coordinates [5]. Following works improve the implicit representation by integrating frequency analysis [20] and modulation-based transformer [48]. Different from these works, we utilize explicit local functions, resulting in advantages in efficiency and interpretability.

Efficient super-resolution with look-up table. A look-up table is composed of index-value pairs, which can be efficiently retrieved through memory access. It is widely applied in the image signal processing pipeline [18, 50]. Recently, SR-LUT has been introduced to accelerate the inference of a fixed-scale super-resolution network by traversing all possible low-resolution (LR) patches, pre-computing all corresponding high-resolution (HR) patches, and saving them as index-value pairs [15]. At inference, the computations in the super-resolution network are replaced with retrieving values from LUT, leading to inference acceleration. Different from existing LUT-based methods [15, 22, 23, 29], which adopt different groups of LUTs under different upsampling scales, our LUTs store the same group of hyper-parameters across arbitrary transformations, thus achieving continuous resampling at high efficiency.

3. Learning Resampling Function

3.1. Formulation

Typically, resampling through interpolation can be implemented in the following steps.

① *Obtain relative offsets*: the target coordinates after transformation, such as upsampling, are projected back to the coordinate space of the input image, and the relative spatial offsets between target pixels and source pixels in their support patches are obtained.

② *Predict resampling weights*: based on the relative spatial offsets, the resampling weights, *i.e.*, resampling kernels, are predicted for each pixel in the support patch.

③ *Aggregate pixels*: the source pixels are aggregated through weighted summation to obtain the target pixel value. The above process can be formulated as

$$\hat{I}_q = \sum_{p \in \Omega} W_{p \to q} I_p \tag{1}$$

where $\hat{I}_q$ is the interpolated pixel value at the target coordinate, I_p the pixel value at the source coordinate, Ω the support patch, and $W_{p \to q}$ the weight from p to q.

As illustrated in Fig. 3(a), an isotropic resampling function $\Phi(\cdot)$, *e.g.* `cubic` in bicubic interpolation, is assumed to predict resampling weights from relative offsets, which can be formulated as

$$W_{p \to q} = \Phi(d_{p \to q}) \tag{2}$$

where $d_{p \to q}$ is the relative offset from p to q, which can be obtained based on the geometric transformation $\mathcal{T}$. Thanks to the continuity of the resampling function Φ, arbitrary transformations from I_{in} to I_{out} can be achieved. The assumption of local continuity contributes to the versatility and high efficiency of interpolation. But the spatially-invariant resampling function leads to blurry results due to the ignorance of local structures.

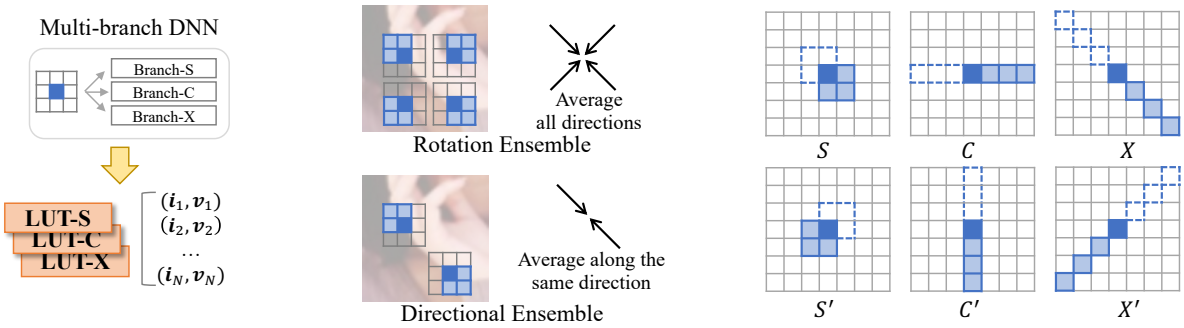
3.2. The Steerable Resampling Function

In this work, we propose a novel method of learning resampling function that adapts the resampling functions to local structures in a data-driven way. Different from the *fixed* resampling function in interpolation, we assume a kind of *steerable* resampling function Φ_Θ, parameterized by Θ. Specifically, we utilize the anisotropic gaussian,

$$\Phi_{\mathbf{\Sigma}}(\boldsymbol{x} - \boldsymbol{\mu}) = \frac{1}{2\pi |\mathbf{\Sigma}|^{\frac{1}{2}}} exp\{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \mathbf{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})\} \tag{3}$$

where $(\boldsymbol{x} - \boldsymbol{\mu})$ are the relative offsets between the target and source pixels, $|\cdot|$ denotes the determinant of the covariance matrix $\mathbf{\Sigma}$. We parameterize $\mathbf{\Sigma}$ as the following,

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_X^2 & \rho \sigma_X \sigma_Y \\ \rho \sigma_X \sigma_Y & \sigma_Y^2 \end{pmatrix} \tag{4}$$



(a) LUT acceleration of DNN (b) Ensemble strategy (c) Edge-sensitive indexing patterns

Figure 5. LUT acceleration of DNN and our adaptations. (a) A learned neural network can be accelerated by traversing all possible inputs i_*, pre-computing all corresponding outputs v_*, and saving them to LUTs [15,22,23,29]. (b) In the proposed directional ensemble strategy, only predictions along the same direction are averaged, instead of all directions in rotation ensemble [15]. (c) We introduce edge-sensitive indexing patterns to better capture edge orientations. The pixels covered by directional ensemble are depicted with dashed boxes.

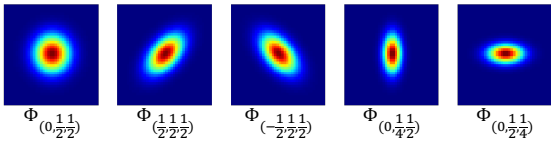


Figure 4. Visualization of parameterized resampling functions.

where ρ, σ_X, and σ_Y are hyper-parameters, and thus the resampling function becomes $\Phi_{(\rho,\sigma_X,\sigma_Y)}$. From the perspective of statistics, ρ can be interpreted as the correlation between 2D variables and σ_X, σ_Y the standard deviations.

As illustrated in Fig. 4, our resampling function is steerable, where different orientations and shapes can be obtained by tuning hyper-parameters $(\rho, \sigma_X, \sigma_Y)$, showing its modeling capacity for a variety of local structures. Other non-steerable functions, such as Keys `cubic` [16] are also applicable as resampling functions. Details of the comparison are provided in Table 3 and Fig. 10.

But, direct optimization of the above hyper-parameters leads to unstable gradients and divergence. Thus, we modify the formulation by omitting the determinant multiplier and predicting a group of $(\rho, \frac{1}{\sigma_X}, \frac{1}{\sigma_Y})$ for stable training. In the following section, we introduce the learning process of these hyper-parameters and our efficient implementation.

3.3. Learning with DNN and LUT Acceleration

As shown in Fig. 3(b), we adopt a deep neural network to learn structural priors from an external dataset for predicting hyper-parameters Θ in resampling functions. The training loss function can be formulated as

$$\mathcal{L}_q = ||I_q - \sum_{p\in\Omega} \Phi_\Theta(d_{p\to q}) I_p||^2, \quad \Theta = f(\mathcal{N}_p), \qquad (5)$$

where I_q is the ground truth pixel value at location q, $f(\cdot)$ is a trainable DNN, and $\mathcal{N}_p$ denotes the surrounding pixels of I_p (not necessarily the same as the support patch Ω).

Furthermore, to match the efficiency of interpolation, we accelerate the inference of the trained DNN by adopting look-up tables [15, 22, 23, 29]. As shown in Fig. 5(a), for each pair in the LUT to accelerate DNN, its index $\boldsymbol{i}_*$ is a combination of pixels (*i.e.*, $\mathcal{N}_p$), and its value $\boldsymbol{v}_*$ is a group of corresponding hyper-parameters for that pixel combination (*i.e.*, Θ). This way, hyper-parameters can be retrieved directly from the saved values in LUTs, skipping computations in DNN and thus resulting in high efficiency.

Different from existing LUT-based methods, whose LUT values are image pixels, our LUTs store hyper-parameters that reflect structural characteristics. Thus, to better extract structural priors, we propose the following adaptations.

Directional ensemble strategy. We propose a directional ensemble (DE) strategy to replace the rotation ensemble (RE) strategy in existing LUT-based methods [15, 22, 23]. As illustrated in Fig. 5(b), in RE, the predictions are averaged across all directions, while the proposed DE strategy only ensembles the predictions with the same direction (*i.e.*, 180° rotational symmetry, instead of 90° in RE). This enables the learning of ρ, which determines the orientation of the steerable resampling function (see Table 5 and Fig. 11).

Edge-sensitive indexing patterns. As illustrated in Fig. 5(c), we include patterns "C" and "X", alongside the default "S" pattern in SR-LUT to better capture edges of different orientations. For example, The "C" and "C'" patterns are sensitive to vertical and horizontal edges, respectively. We validate their effectiveness in Table 5. Correspondingly, as shown in Fig. 5(a), our DNN follows a multi-branch design, and each branch is accelerated by a LUT.

Pre-filtering stage. Inspired by DNN-based resampling methods [12, 43, 48], we pre-filter the image with a neural network, which can also be accelerated by LUTs with the proposed edge-sensitive indexing patterns. This pre-filtering stage helps enhance image content and contributes to better performance (see Table 5 and Fig. 11).

Method	Set5				Set14				BSDS100				Urban100				Manga109			
	$\frac{\times1.5}{\times1.5}$	$\frac{\times1.5}{\times2.0}$	$\frac{\times2.0}{\times2.0}$	$\frac{\times2.0}{\times2.4}$	$\frac{\times1.5}{\times1.5}$	$\frac{\times1.5}{\times2.0}$	$\frac{\times2.0}{\times2.0}$	$\frac{\times2.0}{\times2.4}$	$\frac{\times1.5}{\times1.5}$	$\frac{\times1.5}{\times2.0}$	$\frac{\times2.0}{\times2.0}$	$\frac{\times2.0}{\times2.4}$	$\frac{\times1.5}{\times1.5}$	$\frac{\times1.5}{\times2.0}$	$\frac{\times2.0}{\times2.0}$	$\frac{\times2.0}{\times2.4}$	$\frac{\times1.5}{\times1.5}$	$\frac{\times1.5}{\times2.0}$	$\frac{\times2.0}{\times2.0}$	$\frac{\times2.0}{\times2.4}$
Nearest	31.34	31.07	30.84	29.63	29.15	28.84	28.57	27.70	28.99	28.72	28.40	27.62	26.21	25.91	25.62	24.78	28.59	28.36	28.14	26.87
Bilinear	34.99	33.19	32.23	31.49	31.68	30.26	29.24	28.70	30.92	29.66	28.67	28.20	28.24	26.91	25.96	25.46	32.45	30.33	29.16	28.28
Bicubic	36.76	34.68	33.64	32.70	33.07	31.45	30.32	29.62	32.14	30.67	29.54	28.93	29.50	27.95	26.87	26.22	34.76	32.13	30.81	29.61
Lanczos2	36.83	34.74	33.70	32.74	33.13	31.50	30.36	29.65	32.19	30.71	29.58	28.95	29.55	28.00	26.91	26.25	34.87	32.22	30.89	29.66
Lanczos3	37.61	35.31	34.23	33.24	33.75	31.97	30.76	30.02	32.74	31.11	29.89	29.23	30.12	28.42	27.25	26.55	36.12	33.06	31.63	30.28
RAISR* [37]	35.50	35.49	35.57	33.38	31.84	31.67	31.71	30.22	30.87	30.68	30.66	29.42	28.77	28.60	28.64	27.01	33.81	33.74	**33.88**	30.61
SR-LUT* [15]	37.74	35.52	34.47	33.45	33.53	31.79	30.59	29.86	33.06	31.40	30.15	29.45	30.24	28.55	27.39	26.69	35.31	32.30	30.96	29.58
LeRF (Ours)	**38.30**	**36.60**	**35.71**	**34.74**	**34.59**	**33.06**	**31.98**	**31.10**	**33.76**	**32.08**	**30.83**	**30.09**	**31.86**	**30.08**	**28.86**	**27.99**	**36.57**	**34.79**	**33.88**	**32.67**
Meta-SR [12]	41.29	-	38.12	-	37.47	-	33.99	-	35.79	-	32.32	-	35.85	-	32.98	-	42.92	-	39.22	-
LIIF [5]	41.22	38.99	38.08	36.99	37.44	35.31	33.96	32.95	35.75	33.68	32.28	31.45	36.70	34.08	32.84	31.70	42.77	40.19	39.13	37.69

Method	Set5				Set14				BSDS100				Urban100				Manga109			
	$\frac{\times2.0}{\times3.0}$	$\frac{\times3.0}{\times3.0}$	$\frac{\times3.0}{\times4.0}$	$\frac{\times4.0}{\times4.0}$	$\frac{\times2.0}{\times3.0}$	$\frac{\times3.0}{\times3.0}$	$\frac{\times3.0}{\times4.0}$	$\frac{\times4.0}{\times4.0}$	$\frac{\times2.0}{\times3.0}$	$\frac{\times3.0}{\times3.0}$	$\frac{\times3.0}{\times4.0}$	$\frac{\times4.0}{\times4.0}$	$\frac{\times2.0}{\times3.0}$	$\frac{\times3.0}{\times3.0}$	$\frac{\times3.0}{\times4.0}$	$\frac{\times4.0}{\times4.0}$	$\frac{\times2.0}{\times3.0}$	$\frac{\times3.0}{\times3.0}$	$\frac{\times3.0}{\times4.0}$	$\frac{\times4.0}{\times4.0}$
Nearest	28.87	27.91	26.88	26.25	27.07	26.08	25.33	24.74	27.12	26.17	25.57	25.03	24.25	23.34	22.68	22.17	26.12	25.04	24.06	23.43
Bilinear	30.43	29.53	28.27	27.55	27.94	27.04	26.16	25.51	27.60	26.77	26.11	25.53	24.81	23.99	23.26	22.68	27.21	26.16	24.95	24.19
Bicubic	31.41	30.39	29.12	28.42	28.70	27.63	26.75	26.09	28.18	27.20	26.53	25.95	25.43	24.45	23.71	23.14	28.20	26.95	25.67	24.90
Lanczos2	31.44	30.41	29.14	28.44	28.72	27.64	26.77	26.10	28.20	27.21	26.54	25.96	25.45	24.47	23.73	23.15	28.23	26.97	25.70	24.92
Lanczos3	31.85	30.79	29.49	28.78	29.04	27.91	27.01	26.31	28.43	27.39	26.70	26.10	25.71	24.68	23.92	23.32	28.70	27.38	26.02	25.21
RAISR* [37]	32.35	31.87	29.91	29.65	29.04	28.62	27.10	26.86	28.24	27.84	26.68	26.42	25.92	25.50	24.08	23.89	29.30	28.73	26.40	26.12
SR-LUT* [15]	32.04	31.00	29.70	29.00	28.88	27.84	26.95	26.30	28.62	27.54	26.84	26.21	25.83	24.78	24.00	23.39	28.03	26.74	25.39	24.72
LeRF (Ours)	**33.17**	**32.02**	**30.86**	**30.15**	**30.06**	**28.84**	**28.05**	**27.35**	**29.15**	**28.00**	**27.31**	**26.70**	**26.90**	**25.68**	**24.88**	**24.23**	**30.86**	**29.48**	**28.10**	**27.25**
Meta-SR [12]	-	34.71	-	32.48	-	30.56	-	28.83	-	29.26	-	27.73	-	28.91	-	26.69	-	34.37	-	31.32
LIIF [5]	35.63	34.59	33.17	32.37	31.68	30.39	29.45	28.65	30.47	29.24	28.42	27.73	30.23	28.80	27.55	26.66	35.73	34.17	32.29	31.19

Table 1. Quantitative comparison in PSNR for arbitrary-scale upsampling. $\frac{\times r_h}{\times r_w}$ denotes upsampling r_h times along the short side and r_w times along the long side. SSIM and LPIPS results are in the supplementary document. * denotes that we combine fixed-scale super-resolution methods with bicubic to achieve arbitrary-scale upsampling. The best and second best results are **highlighted** and underlined.

4. Experiments and Results

4.1. Experimental Settings

Datasets and metrics. We train LeRF on the DIV2K dataset [1], which is widely used in image resampling tasks. We train LeRF on the $\times 4$ downsampled data pairs, and apply the obtained model to resampling for arbitrary transformations. For arbitrary-scale upsampling, we evaluate LeRF with 5 benchmark datasets: Set5, Set14, BSDS100 [30], Urban100 [13], and Manga109 [31]. We select representative symmetric or asymmetric upsampling scales for evaluation and apply bicubic interpolation as the degradation model to obtain the LR images. For performance evaluation, we report PSNR and SSIM [45] for fidelity, and LPIPS [53] for perceptual quality. For efficiency evaluation, we report running time, theoretical multiply-accumulate operations (MACs), and storage requirements to evaluate the performance-efficiency trade-off.

Comparison methods. The main competitors to the proposed method are interpolation methods, including nearest neighbor, bilinear, bicubic, lanczos2, and lanczos3. Furthermore, we combine fixed-scale super-resolution methods with bicubic interpolation as additional baselines for arbitrary-scale upsampling (RAISR* and SR-LUT*).

Method	RunTime (ms)	MACs	Storage Size	PSNR/SSIM (dB)/-
Nearest	11	-	-	24.74/0.6553
Bilinear	31	14.74M	-	25.51/0.6824
Bicubic	126	51.61M	-	26.09/0.7056
Lanczos2	494	110.59M	-	26.10/0.7058
Lanczos3	914	165.89M	-	26.31/0.7130
RAISR* [37]	3,744	-	3.19MB	26.86/0.7357
SR-LUT* [15]	137	53.33M	81.56KB	26.30/0.7256
LeRF (Ours)	110	57.94M	1.67MB	27.35/0.7475
MetaSR [12]	10,260	1.68T	85.59MB	28.83/0.7876
LIIF [5]	67,080	2.54T	255.76MB	28.65/0.7878

Table 2. Efficiency comparison of running time, MACs, storage requirements, and performance (Set14) for producing a 1280×720 HD image through $4\times$ upsampling. For interpolation methods, SR-LUT*, and LeRF, the running time is evaluated on CPU of a mobile phone and averaged across 10 trials, while for RAISR* and DNNs, that is evaluated on CPU of a desktop computer.

Specifically, since RAISR [37] lacks generalization ability across different upsampling scales, we fuse the results of $\times2$, $\times3$, and $\times4$ models by choosing the model with the closest integer upsampling scales (*e.g.*, $\times3$ model for $\frac{\times2.0}{\times3.0}$). We retrain SR-LUT [15] on the mixed-scale data of $\times2$, $\times3$, and $\times4$ downsampled data pairs jointly with bicubic interpolation to achieve arbitrary-scale upsampling. Finally, we

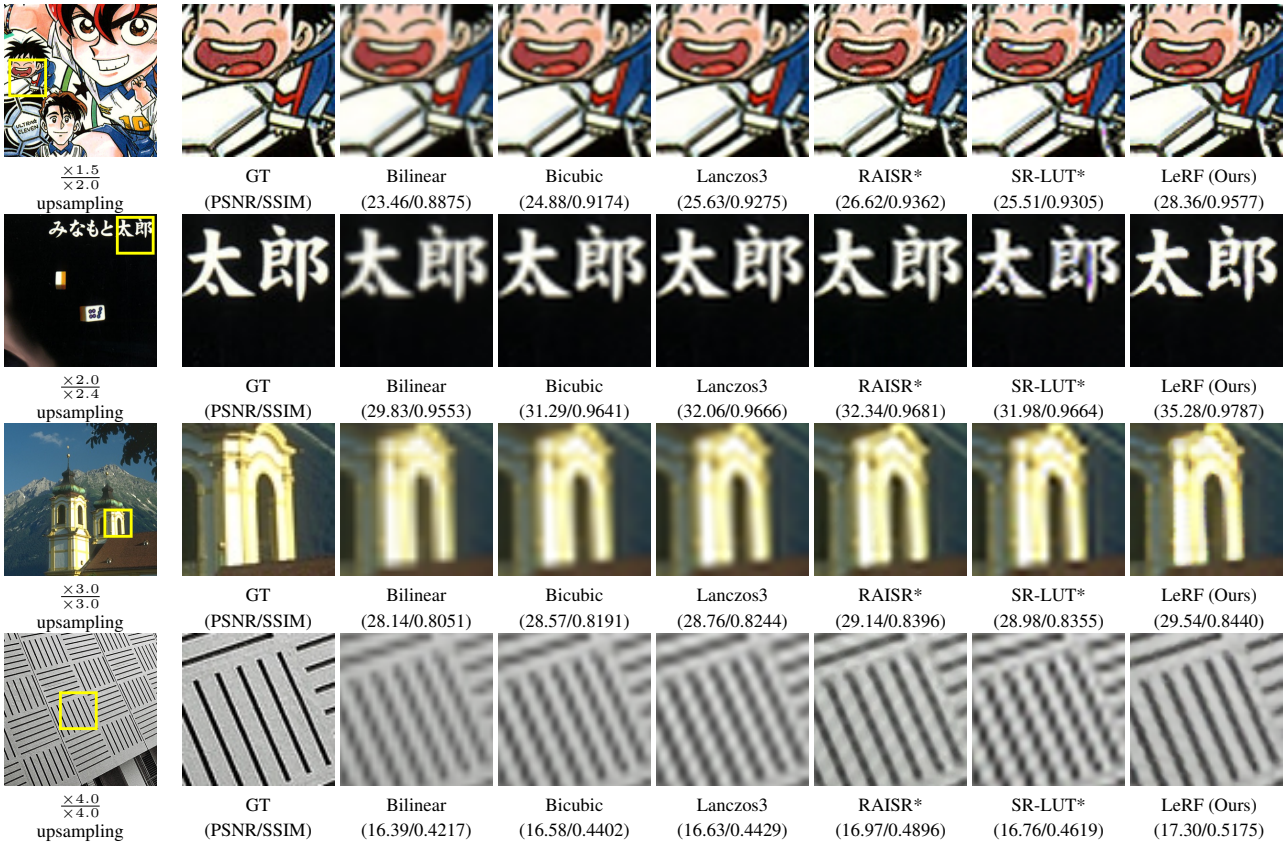


Figure 6. Qualitative comparison for arbitrary-scale upsampling. From top to bottom, the example images are: *UltraEleven* (Manga109), *MukoukizuNoChonbo* (Manga109), *126007* (BSDS100), and *img092* (Ubran100). Best view in color and on screen.

include DNN-based arbitrary-scale super-resolution methods (Meta-SR [12] and LIIF [5]) for reference.

Implementation details. Our method is trained with the Adam optimizer [19] in the cosine annealing schedule [28]. We train LeRF with the MSE loss function for 5×10^4 iterations at a batch size of 32. Except for the indexing patterns, each branch in our DNN keeps the same architecture as the network in SR-LUT. To obtain a fair comparison in running time tests, we implement LeRF and interpolation methods on the Android platform under the same parallel API, with the only difference being the resampling functions and LUT retrieval in our method. Our method is implemented in MindSpore [33] and the code is available at https://gitee.com/mindspore/models/tree/master/research/cv/lerf.

4.2. Performance Evaluation

Quantitative comparison. We list the quantitative comparisons with other methods in Table 1. As can be seen, our method achieves significantly better performance than interpolation methods. For example, it exceeds bicubic interpolation up to 3dB PSNR when upsampling $\frac{\times 2.0}{\times 2.0}$ on the Manga109 dataset.

Qualitative comparison. In Fig. 6, we compare the visual quality of LeRF with interpolation methods, RAISR*, and SR-LUT*. As can be seen, our method obtains better visual quality across various local structures and textures. Specifically, since the resampling functions are adapted to local structures, LeRF is capable of retaining clearer textures and avoiding blurry boundaries.

Visualization of learned resampling functions. We further visualize the intermediate results of the proposed method in Fig. 7. As can be seen, the hyper-parameter ρ clearly distinguishes the orientations of edges (red vs. blue), and σ_X and σ_Y capture the horizontal and vertical lines, respectively. In addition, in the second row of each example, we visualize the resampling functions defined by the predicted hyper-parameters. The shapes of resampling functions are well adapted to corners, flat surfaces, and edges with various orientations. This validates the effectiveness of extracting structural priors in a data-driven way.

4.3. Efficiency Evaluation

To evaluate the efficiency, we conduct running time tests, estimate the theoretical MACs, and report storage requirements for LeRF and comparison methods. We list the de-

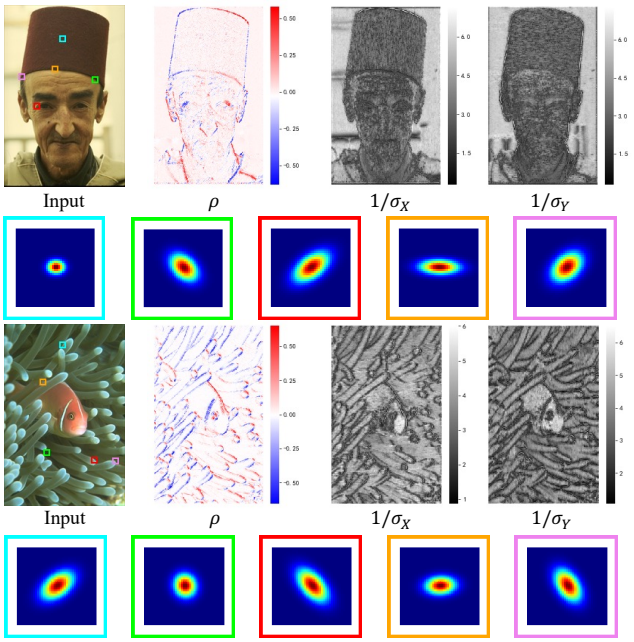


Figure 7. Visualization of the pixel-wise hyper-parameters and the corresponding resampling functions. The shapes of the predicted resampling functions are well adapted to local structures.

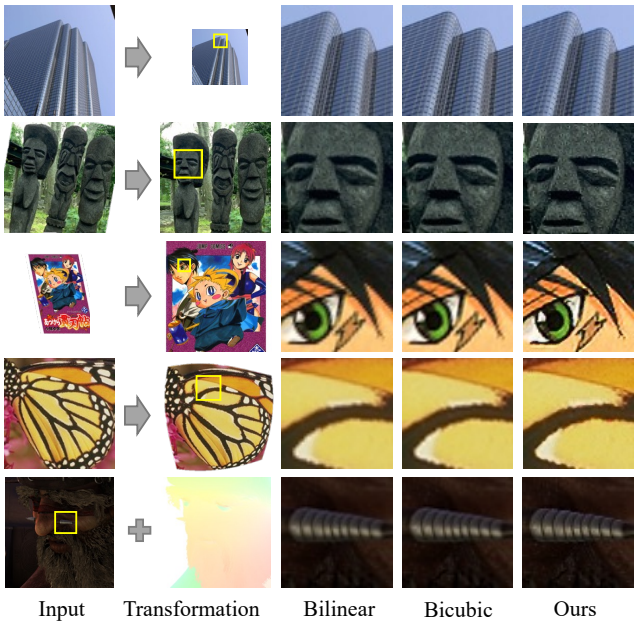


Figure 8. Visual comparison of LeRF with interpolation methods under general homographic transformations (downsampling, rotation, and sheering) and arbitrary warping (according to a barrel-shaped distortion and optical flow).

tailed comparison in Table 2. As can be seen, the proposed LeRF achieves comparable or better running time compared to the popular interpolation methods, showing its role as a superior competitor with significantly better performance. On the other hand, compared to DNN-based methods, LeRF shows a clear advantage in efficiency. Furthermore, our method requires far less extra storage as well as fewer software dependencies, demonstrating its practicality for being deployed on various devices.

4.4. Generalization Evaluation

In Fig. 8, we evaluate the generalization ability of LeRF to arbitrary transformations and compare it with widely applied interpolation methods. As can be seen, the learned resampling functions generalize well to unseen deformations. LeRF generates sharper edges and retains more texture details, leading to more visually pleasing results than popular interpolation methods.

4.5. Ablation Analysis

Analysis of GT-optimized resampling functions. To analyze the optimal shapes of resampling functions across different transformations, we obtain hyper-parameters in resampling functions via per-image optimization. In Fig. 9, we visualize the clustering results on Set14 of these GT-optimized resampling functions across different upsampling scales. As can be seen, the GT-optimized cluster centroids are very similar across different upsampling scales, explaining the generalization ability of our method.

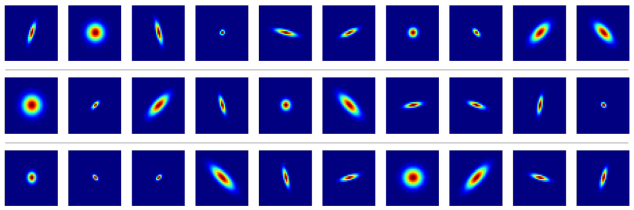


Figure 9. Visualization of GT-optimized resampling function clustering centroids on Set14 for $\times 2$, $\times 3$, and $\times 4$ upsampling, respectively. They show consistency across different upsampling scales.

The effectiveness of the steerable resampling function. We conduct the following ablation experiments on the Set5 benchmark to analyze the design of the steerable resampling function. 1) Fixed gaussian: We freeze the hyper-parameters in the anisotropic gaussian. 2) Learned Keys `cubic`: We utilize another non-steerable resampling function, the isotropic Keys `cubic` function [16], which has only one hyper-parameter that controls the sharpness of the one-dimension piecewise `cubic`. The experimental results are listed in Table 3, and visual results are shown in Fig. 10. As can be seen, non-steerable resampling functions like Keys `cubic` produce zig-zag artifacts and inferior performance, showing the effectiveness of our learned steerable resampling functions.

Analysis of training data. By default, LeRF is trained with $\times 4$ downsampled data pairs. We retrain our method on $\times 2$ data pairs and mixed data pairs with both symmetric and

	MACs	Storage	×2	×3	×4
Ours	57.94M	1.67MB	35.71	32.02	30.15
fixed gaussian $\Phi_{(0,1,1)}$	45.04M	244.69KB	30.75	27.70	26.31
learned Keys `cubic`	126.94M	489.38KB	34.68	31.17	29.08

Table 3. Ablation experiments on Set5 on the the design of the steerable resampling function.

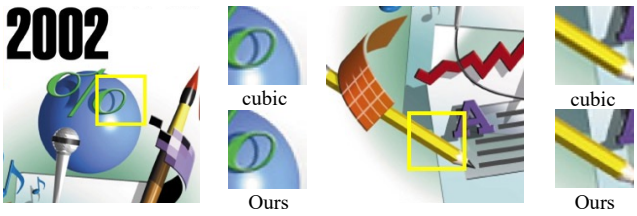


Figure 10. The non-steerable Keys `cubic` function produces artifacts along diagonal edges.

	MACs	Storage	×2	×3	×4
Ours (×4 data)	57.94M	1.67MB	35.71	32.02	30.15
×2 data training	57.94M	1.67MB	35.67	32.11	29.91
mixed-scale training	57.94M	1.67MB	35.67	32.26	30.06

Table 4. Ablation experiments on Set5 on data pairs for training.

asymmetric upsampling scales. As listed in Table 4, training on mixed data pairs yields similar performance, showing the robustness and generalization ability of our method, which is consistent with the clustering results in Fig. 9.

The effectiveness of the LUT acceleration and the proposed adaptations. As listed in Table 5, we conduct the following ablation experiments to analyze the design of the LUT acceleration and our adaptations. 1) Without LUT acceleration: Compared with the learned network, the LUT acceleration saves a lot of computations at a very small performance cost. 2) Without DE: The visualization of the learned hyper-parameter ρ in Fig. 11 shows that RE makes an isotropic assumption and thus lacks the ability to learn edge orientations, resulting in a performance drop. 3) Without edge-sensitive patterns: LeRF with "SCX" indexing patterns outperforms the variant with only the default "S" pattern, showing the ability of edge-sensitive indexing patterns to better capture edge structures. 4) Without pre-filtering or pre-filtering only: As shown in Fig. 11 and Table 4, the pre-filtering stage enhances edges and contributes to better performance, and pre-filtering only is suboptimal due to the lack of adpativity of resampling functions.

4.6. Limitation

In most cases, our method produces better results compared with interpolation. But, it may fail under certain circumstances. In Fig. 12, we show its limitation in downsampling, where aliasing artifacts appear in regions with highly dense textures. It is probably caused by a lack of training on downsampled data. Moreover, replacing the pre-filtering stage with anti-aliasing filters may mitigate this problem [6, 7].

	MACs	Storage	×2	×3	×4
Ours	57.94M	1.67MB	35.71	32.02	30.15
w/o LUT acceleration	21.75G	1.59MB	36.11	32.23	30.28
w/o DE	57.94M	978.76KB	34.44	31.50	29.47
w/o "CX" pattern	49.65M	326.25KB	34.90	31.79	29.70
w/o pre-filtering	53.45M	1.43MB	34.43	31.41	29.33
pre-filtering only	56.10M	244.69KB	34.69	31.16	29.15

Table 5. Ablation experiments on Set5 on the LUT acceleration and our adaptations.

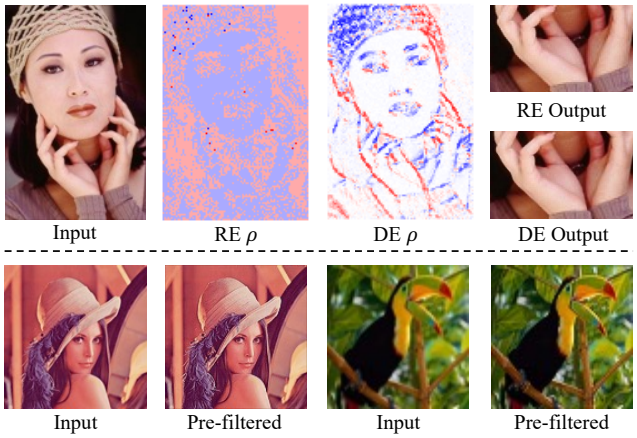


Figure 11. Upper: Replacing RE with DE enables the learning of edge orientations. Lower: The pre-filtering stage enhances edges.

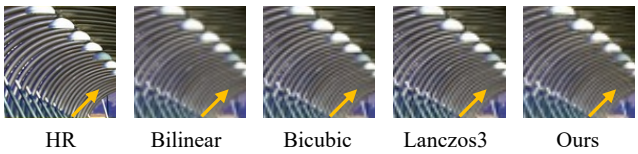


Figure 12. Limitation. LeRF produces aliasing artifacts for image downsampling in regions with highly dense textures.

5. Conclusion Remarks

In this work, we propose LeRF, a novel method for continuous resampling by integrating learned structural priors into the steerable resampling function. We show its superior performance, high efficiency, and versatility for arbitrary transformations. In the future, we plan to extend this idea to other modalities, *e.g.*, video and point clouds.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China under Grants 62131003 and 62021001. We acknowledge the support of MindSpore.

References

[1] Eirikur Agustsson and Radu Timofte. NTIRE 2017 challenge on single image super-resolution: Dataset and study. In *CVPR Workshops*, 2017. 5

[2] Jan P. Allebach and Ping Wah Wong. Edge-directed interpolation. In *ICIP*, 1996. 2

[3] Chang Chen, Xinmei Tian, Feng Wu, and Zhiwei Xiong. UDNet: Up-down network for compact and efficient feature representation in image super-resolution. In *ICCV Workshops*, 2017. 2

[4] Chang Chen, Zhiwei Xiong, Xinmei Tian, Zheng-Jun Zha, and Feng Wu. Camera lens super-resolution. In *CVPR*, 2019. 2

[5] Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. In *CVPR*, 2021. 1, 2, 3, 5, 6

[6] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Trans. Graph.*, 5(1):51–72, 1986. 8

[7] Franklin C. Crow. The aliasing problem in computer-generated shaded images. *Commun. ACM*, 20(11):799–805, 1977. 8

[8] Neil Anthony Dodgson. Image resampling. Technical report, University of Cambridge, Computer Laboratory, 1992. 1

[9] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, 2014. 1, 2

[10] Irvine C. Gardner and Francis E. Washer. Lenses of extremely wide angle for airplane mapping. *J. Opt. Soc. Am.*, 38(5):421–431, 1948. 1

[11] Pascal Getreuer, Ignacio Garcia-Dorado, John Isidoro, Sungjoon Choi, Frank Ong, and Peyman Milanfar. BLADE: filter learning for general purpose computational photography. In *ICCP*, 2018. 2

[12] Xuecai Hu, Haoyuan Mu, Xiangyu Zhang, Zilei Wang, Tieniu Tan, and Jian Sun. Meta-SR: A magnification-arbitrary network for super-resolution. In *CVPR*, 2019. 1, 2, 4, 5, 6

[13] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR*, 2015. 5

[14] Xu Jia, Hong Chang, and Tinne Tuytelaars. Super-resolution with deep adaptive image resampling. *arxiv*, 1712.06463, 2017. 2

[15] Younghyun Jo and Seon Joo Kim. Practical single-image super-resolution using look-up table. In *CVPR*, 2021. 2, 3, 4, 5

[16] Robert G. Keys. Cubic convolution interpolation for digital image processing. *IEEE Trans. Acoust.*, 29:1153–1160, 1981. 1, 2, 4, 7

[17] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, 2016. 1, 2

[18] Seon Joo Kim, Hai Ting Lin, Zheng Lu, Sabine Süsstrunk, Stephen Lin, and Michael S. Brown. A new in-camera imaging model for color computer vision and its application. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(12):2289–2302, 2012. 3

[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 6

[20] Jaewon Lee and Kyong Hwan Jin. Local texture estimator for implicit representation function. In *CVPR*, 2022. 3

[21] Yeon Ju Lee and Jungho Yoon. Nonlinear image upsampling method based on radial basis function interpolation. *IEEE Trans. Image Process.*, 19(10):2682–2692, 2010. 2

[22] Jiacheng Li, Chang Chen, Zhen Cheng, and Zhiwei Xiong. MuLUT: Cooperating multiple look-up tables for efficient image super-resolution. In *ECCV*, 2022. 2, 3, 4

[23] Jiacheng Li, Chang Chen, Zhen Cheng, and Zhiwei Xiong. Toward DNN of LUTs: Learning efficient image restoration with multiple look-up tables. *arxiv*, 2303.14506, 2023. 2, 3, 4

[24] Xin Li and Michael T. Orchard. New edge-directed interpolation. *IEEE Trans. Image Process.*, 10(10):1521–1527, 2001. 2

[25] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. SwinIR: Image restoration using swin transformer. In *ICCV Workshops*, 2021. 2

[26] Jingyun Liang, Kai Zhang, Shuhang Gu, Luc Van Gool, and Radu Timofte. Flow-based kernel prior with application to blind super-resolution. In *CVPR*, 2021. 2

[27] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR Workshops*, 2017. 1, 2

[28] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *ICLR*, 2017. 6

[29] Cheng Ma, Jingyi Zhang, Jie Zhou, and Jiwen Lu. Learning series-parallel lookup tables for efficient image super-resolution. In *ECCV*, 2022. 2, 3, 4

[30] David R. Martin, Charless C. Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, 2001. 5

[31] Yusuke Matsui, Kota Ito, Yuji Aramaki, Azuma Fujimoto, Toru Ogawa, Toshihiko Yamasaki, and Kiyoharu Aizawa. Sketch-based manga retrieval using manga109 dataset. *Multim. Tools Appl.*, 76(20):21811–21838, 2017. 5

[32] Ben Mildenhall, Jonathan T. Barron, Jiawen Chen, Dillon Sharlet, Ren Ng, and Robert Carroll. Burst denoising with kernel prediction networks. In *CVPR*, 2018. 2

[33] MindSpore. https://www.mindspore.cn, 2023. 6

[34] Don P. Mitchell and Arun N. Netravali. Reconstruction filters in computer-graphics. In *SIGGRAPH*, 1988. 2

[35] Marta Mrak, Mislav Grgic, and Murat Kunt. *High-Quality Visual Experience: Creation, Processing and Interactivity of High-Resolution and High-Dimensional Video Signals*. Signals and Communication Technology. Springer Berlin Heidelberg, 2016. 1

[36] Zhihong Pan, Baopu Li, Dongliang He, Mingde Yao, Wenhao Wu, Tianwei Lin, Xin Li, and Errui Ding. Towards bidirectional arbitrary image rescaling: Joint optimization and cycle idempotence. In *CVPR*, 2022. 2

[37] Yaniv Romano, John Isidoro, and Peyman Milanfar. RAISR: rapid and accurate image super resolution. *IEEE Trans. Computational Imaging*, 3(1):110–125, 2017. 2, 5

[38] Thomas G. Smith and George Lucas. *Industrial Light & Magic: The Art of Special Effects*. Virgin, 1986. 1

[39] Sanghyun Son and Kyoung Mu Lee. SRWarp: Generalized image super-resolution under arbitrary transformation. In *CVPR*, 2021. 1, 2

[40] Wanjie Sun and Zhenzhong Chen. Learned image downscaling for upscaling using content adaptive resampler. *IEEE Trans. Image Process.*, 29:4027–4040, 2020. 1, 2

[41] Hiroyuki Takeda, Sina Farsiu, and Peyman Milanfar. Kernel regression for image processing and reconstruction. *IEEE Trans. Image Process.*, 16(2):349–366, 2007. 2

[42] Hossein Talebi and Peyman Milanfar. Learning to resize images for computer vision tasks. In *ICCV*, 2021. 2

[43] Longguang Wang, Yingqian Wang, Zaiping Lin, Jungang Yang, Wei An, and Yulan Guo. Learning A single network for scale-arbitrary super-resolution. In *ICCV*, 2021. 1, 2, 4

[44] Qing Wang and Rabab Kreidieh Ward. A new orientation-adaptive interpolation method. *IEEE Trans. Image Process.*, 16(4):889–900, 2007. 2

[45] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.*, 13(4):600–612, 2004. 5

[46] Bartlomiej Wronski, Ignacio Garcia-Dorado, Manfred Ernst, Damien Kelly, Michael Krainin, Chia-Kai Liang, Marc Levoy, and Peyman Milanfar. Handheld multi-frame super-resolution. *ACM Trans. Graph.*, 38(4):28:1–28:18, 2019. 2

[47] Zeyu Xiao, Xueyang Fu, Jie Huang, Zhen Cheng, and Zhiwei Xiong. Space-time distillation for video super-resolution. In *CVPR*, 2021. 2

[48] Jingyu Yang, Sheng Shen, Huanjing Yue, and Kun Li. Implicit transformer network for screen content image continuous super-resolution. In *NeurIPS*, 2021. 1, 3, 4

[49] Mingde Yao, Dongliang He, Xin Li, Zhihong Pan, and Zhiwei Xiong. Bidirectional translation between uhd-hdr and hd-sdr videos. *IEEE Trans. Multimedia*, 2023. 2

[50] Hui Zeng, Jianrui Cai, Lida Li, Zisheng Cao, and Lei Zhang. Learning image-adaptive 3d lookup tables for high performance photo enhancement in real-time. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(4):2058–2073, 2022. 3

[51] Kaibing Zhang, Xinbo Gao, Dacheng Tao, and Xuelong Li. Single image super-resolution with non-local means and steering kernel regression. *IEEE Trans. Image Process.*, 21(11):4544–4556, 2012. 2

[52] Lei Zhang and Xiaolin Wu. An edge-guided image interpolation algorithm via directional filtering and data fusion. *IEEE Trans. Image Process.*, 15(8):2226–2238, 2006. 2

[53] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 5

[54] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *ECCV*, 2018. 2