GRAPH ALGORITHM UNROLLING WITH DOUGLAS-RACHFORD ITERATIONS FOR IMAGE INTERPOLATION WITH GUARANTEED INITIALIZATION

Xue Zhang⋆*, Bingshuo Hu*⋆*, Gene Cheung*†

⋆College of Computer Science and Engineering, Shandong University of Science and Technology, China
†Dept of EECS, York University, Canada

ABSTRACT

Conventional deep neural nets (DNNs) initialize network parameters at random and then optimize each one via stochastic gradient descent (SGD), resulting in substantial risk of poor-performing local minima. Focusing on the image interpolation problem and leveraging a recent theorem that maps a (pseudo-)linear interpolator $\mathbf{\Theta}$ to a directed graph filter that is a solution to a MAP problem regularized with a graph shift variation (GSV) prior, we first initialize a directed graph adjacency matrix $\mathbf{A}$ based on a known interpolator $\mathbf{\Theta}$, establishing a baseline performance. Then, towards further gain, we learn perturbation matrices $\mathbf{P}$ and $\mathbf{P}^{(2)}$ from data to augment $\mathbf{A}$, whose restoration effects are implemented via Douglas-Rachford (DR) iterations, which we unroll into a lightweight interpretable neural net. Experimental results demonstrate state-of-the-art image interpolation results, while drastically reducing network parameters.

***Index Terms*—** Image interpolation, algorithm unrolling, graph signal processing, convex optimization

1. INTRODUCTION

Image interpolation—a well-studied image restoration task—computes missing pixel values at targeted 2D grid locations given observed neighboring image pixels. While early methods include model-based ones such as linear and bicubic interpolations [1], recent methods are dominated by deep-learning-based models such as SwinIR [2] and Restormer [3], driven by powerful off-the-shelf neural architectures, such as *convolutional neural nets* (CNNs) [4] and transformers [5]. However, these models require huge numbers of network parameters, leading to large training and storage costs. Moreover, initialization of network parameters are typically done in a random way, followed by local tuning via *stochastic gradient descent* (SGD) [6], resulting in a high risk of converging to poor-performing local minima.

An alternative to "black box" network architectures is *algorithm unrolling* [7], where iterations of an iterative algorithm minimizing a well-defined optimization objective are implemented as a sequence of neural layers to compose a feed-forward network, and then selected parameters are subsequently optimized end-to-end via back-propagation. As one notable example, [8] unrolled an algorithm minimizing a sparse rate reduction (SRR) objective, resulting in a "white box" transformer-like neural net that is 100% mathematically interpretable. Orthogonally, researchers in *graph signal processing* (GSP) [9, 10] recently unrolled graph algorithms minimizing smoothness priors such as *graph Laplacian regularizer* (GLR) [11] and *graph total variation* (GTV) [12] into lightweight transformers for image interpolation [13]. The key insight is that *a graph learning module with normalization is akin to the classical self-attention mechanism* [14], and thus a neural net constructed from graph algorithm unrolling is a transformer. However, parameter initialization for feature representation is still done randomly.

In this paper, leveraging a recent theorem [15] that maps a (pesudo-)linear interpolator $\mathbf{\Theta}$ to a directed graph filter that is a solution to a *maximum a posteriori* (MAP) problem using *graph shift variation* (GSV) [16] as signal prior, we first initialize a neural net, unrolled from an iterative graph algorithm solving the MAP problem, to a known interpolator $\mathbf{\Theta}$, so that it has guaranteed baseline performance. Then, we augment the initialized adjacency matrix $\mathbf{A}$ in the graph filter in a data-driven manner using two perturbation matrices $\mathbf{P}$ and $\mathbf{P}^{(2)}$ representing two graphs: i) a *directed* graph $\mathcal{G}^d$ connecting interpolated pixels to observed pixels to improve interpolation, and ii) an *undirected* graph $\mathcal{G}^u$ interconnecting interpolated pixels for further denoising. Finally, we implement the restoration effects of $\mathbf{P}$ and $\mathbf{P}^{(2)}$ via *Douglas-Rachford* (DR) iterations [17], so that they can be data-tuned individually. Experiments show that our unrolled transformer-like neural net achieves state-of-the-art (SOTA) image interpolation performance compared to SwinIR [2], Restormer [3] and graph-based uGTV [13], while drastically reducing network parameters (*e.g.*, 1% of Restormer [3]).

2. PRELIMINARIES

2.1. Graph Shift Variation

A smooth (consistent) signal $\mathbf{x}$ with respect to (w.r.t.) a graph $\mathcal{G}$ can be mathematically described in many ways, *e.g.*, *graph Laplacian regularizer* (GLR) [11], *graph total variation* (GTV) [12]. Here, we focus on *graph shift variation* (GSV) [16]:

$$R(\mathbf{x}) = \|\mathbf{x} - \mathbf{A}\mathbf{x}\|_2^2. \tag{1}$$

Using $\mathbf{A}$ as a *graph shift operator* (GSO) [9]—*e.g.*, row-stochastic adjacency matrix $\mathbf{D}^{-1}\mathbf{W}$—(1) states that signal $\mathbf{x}$ and its shifted version $\mathbf{A}\mathbf{x}$ should be similar in an ℓ_2-norm sense. Unlike GLR, GSV can be used for directed graphs also.

2.2. Interpolation Theorem

We first review the interpolator theorem in [15]. Consider a linear interpolator $\mathbf{\Theta} \in \mathbb{R}^{N \times M}$ that interpolates N new pixels from M original pixels $\mathbf{y} \in \mathbb{R}^M$. Mathematically, we write

$$\mathbf{x} = \begin{bmatrix} \mathbf{I}_M \\ \mathbf{\Theta} \end{bmatrix} \mathbf{y} \tag{2}$$

where $\mathbf{x} = [\mathbf{x}_M; \mathbf{x}_N] \in \mathbb{R}^{M+N}$ is the length-$(M+N)$ target signal that retains the original M pixels, *i.e.*, $\mathbf{x}_M = \mathbf{y}$. Note that (2)

Corresponding author: Gene Cheung (genec@yorku.ca).

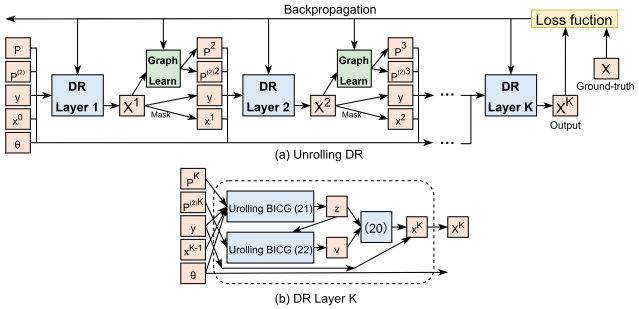

Fig. 1: Network Architecture from Unrolling of Douglas-Rachford iterations into neural layers. Note that a graph learning module (self-attention mechanism) is periodically inserted, as done in [13].

includes *pseudo-linear* operator $\mathbf{\Theta}(\mathbf{y})$, where interpolation weights are first computed as (possibly non-linear) functions of input $\mathbf{y}$, then interpolated pixels are computed via matrix-vector multiplication.

Consider next a graph filter that is a solution to a MAP problem with GSV (1) as signal prior. Denote by $\mathbf{H} = [\mathbf{I}_M \;\; \mathbf{0}_{M,N}] \in \mathbb{R}^{M\times(M+N)}$ a *sampling matrix* that selects the first M original samples from vector $\mathbf{x}$, where $\mathbf{0}_{M,N}$ is a $M \times N$ zero matrix. Denote by $\mathbf{A} \in \mathbb{R}^{(M+N)\times(M+N)}$ an *asymmetric* adjacency matrix specifying directional edges in a directed graph $\mathcal{G}^d$. Specifically, $\mathbf{A}$ describes edges only from N new pixels back to the M original pixels, *i.e.*,

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_{M,M} & \mathbf{A}_{M,N} \\ \mathbf{0}_{N,M} & \mathbf{0}_{N,N} \end{bmatrix}. \tag{3}$$

We now write a MAP objective using GSV (1) as prior:

$$\min_{\mathbf{x}} ||\mathbf{y} - \mathbf{H}\mathbf{x}||_2^2 + \mu||\mathbf{H}(\mathbf{x} - \mathbf{A}\mathbf{x})||_2^2. \tag{4}$$

The two terms in objective (4) are both convex for any $\mathbf{H}$ and $\mathbf{A}$. To obtain solution $\mathbf{x}^*$ to (4), we take the derivative w.r.t. $\mathbf{x}$ and set it to 0, resulting in

$$\underbrace{\Big(\mathbf{H}^\top\mathbf{H} + \mu(\mathbf{I}-\mathbf{A})^\top\mathbf{H}^\top\mathbf{H}(\mathbf{I}-\mathbf{A})\Big)}_{\mathbf{C}}\mathbf{x}^* = \mathbf{H}^\top\mathbf{y}. \tag{5}$$

Given the definitions of $\mathbf{H}$ and $\mathbf{A}$, $\mathbf{C}$ has a unique structure:

$$\mathbf{C} = \begin{bmatrix} (1+\mu)\mathbf{I}_M & -\mu\mathbf{A}_{M,N} \\ -\mu(\mathbf{A}_{M,N})^\top & \mu\mathbf{A}^2_{N,N} \end{bmatrix}, \tag{6}$$

where $\mathbf{A}^2_{N,N} \triangleq (\mathbf{A}_{M,N})^\top\mathbf{A}_{M,N}$. The solution $\mathbf{x}^*$ to (5) is

$$\mathbf{x}^* = \mathbf{C}^{-1}\mathbf{H}^\top\mathbf{y} = \begin{bmatrix} \mathbf{I}_M \\ \mathbf{A}^{-1}_{M,N} \end{bmatrix}\mathbf{y} \tag{7}$$

where we assume $\mathbf{A}_{M,N}$ is square and invertible, *i.e.*, $M = N$ and $\mathbf{A}_{M,N}$ contains no zero eigenvalues.

We restate the interpolator theorem in [15] to connect a linear interpolator $[\mathbf{I}_M; \mathbf{\Theta}]$ (2) to a corresponding graph filter that is a solution to the MAP problem (4) using GSV as prior.

Theorem 1 *Interpolator* $[\mathbf{I}_M; \mathbf{\Theta}]$ *is the solution filter to the MAP problem* (4) *if* $M = N$, $\mathbf{\Theta}$ *is invertible, and* $\mathbf{A}_{M,N} = \mathbf{\Theta}^{-1}$.

3. PROBLEM FORMULATION

3.1. Feed-Forward Network Construction

Theorem 1 states that, under mild conditions, there is a one-to-one mapping from an interpolator $[\mathbf{I}_M; \mathbf{\Theta}]$ to a directed graph filter that is a solution to MAP problem (4). Leveraging Theorem 1, our goal is to first unroll a graph algorithm minimizing (4) to a neural net initialized to $\mathbf{\Theta}$, then further improve performance via data-driven parameter learning.

Given a linear interpolator $\mathbf{\Theta}$, we leverage Theorem 1 and initialize $\mathbf{A}_{M,N} = \mathbf{\Theta}^{-1}$. We then perturb $\mathbf{A}$ to $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{P}$ for perturbation matrix $\mathbf{P}$ representing a *directed* graph $\mathcal{G}^d$ connecting interpolated pixels to original ones:

$$\mathbf{P} = \begin{bmatrix} \mathbf{0}_{M,M} & \mathbf{P}_{M,N} \\ \mathbf{0}_{N,M} & \mathbf{0}_{N,N} \end{bmatrix}. \tag{8}$$

From (7), it is clear that the new interpolated signal $\mathbf{x}^*$ is

$$\mathbf{x}^* = \begin{bmatrix} \mathbf{I}_M \\ \tilde{\mathbf{A}}^{-1}_{M,N} \end{bmatrix}\mathbf{y} = \begin{bmatrix} \mathbf{I}_M \\ (\mathbf{A}_{M,N} + \mathbf{P}_{M,N})^{-1} \end{bmatrix}\mathbf{y}. \tag{9}$$

However, (9) involves a computation-intensive matrix inversion.

3.1.1. Computing Interpolated Pixels

Given interpolator $\mathbf{\Theta}$, to compute interpolated pixels $\mathbf{x}^*_N \in \mathbb{R}^N$ *without* matrix inversion, we rewrite (9) as

$$(\mathbf{A}_{M,N} + \mathbf{P}_{M,N})\mathbf{x}^*_N = \mathbf{y} \tag{10}$$

$$\mathbf{x}^*_N + \underbrace{\mathbf{A}^{-1}_{M,N}}_{\mathbf{\Theta}}\mathbf{P}_{M,N}\mathbf{x}^*_N = \underbrace{\mathbf{A}^{-1}_{M,N}}_{\mathbf{\Theta}}\mathbf{y} \tag{11}$$

$$(\mathbf{I}_N + \mathbf{\Theta}\mathbf{P}_{M,N})\mathbf{x}^*_N = \mathbf{\Theta}\,\mathbf{y}. \tag{12}$$

(12) is a linear system, and though the coefficient matrix $\mathbf{I}_M + \mathbf{\Theta}\mathbf{P}_{M,N}$ is not symmetric in general, $\mathbf{x}^*_N$ can nonetheless be computed efficiently via *biconjugate gradient* (BiCG) [18]—an iterative algorithm similar to *conjugate gradient* (CG) [19]—extended for asymmetric coefficient matrices.

We unroll BiCG iterations into neural layers to compose a feed-forward network as done in [13], so that parameters for BiCG and $\mathbf{P}_{M,N}$ can be optimized in a data-driven manner. Specifically, each BiCG iteration has two parameters α and β that correspond to step size and momentum term. $\mathbf{P}_{M,N}$ specifies an N-node directed graph $\mathcal{G}^d$, where *signed* edge weights $w_{i,j} \in [-1, 1]$ can be computed as a function of *feature distance* $d_{i,j}$:

$$w_{i,j} = \frac{-2}{1 + e^{-\left(d_{i,j} - d^*\right)}} + 1, \qquad d_{i,j} = \mathbf{f}_j^\top\mathbf{M}\mathbf{f}_i. \tag{13}$$

$\mathbf{f}_i \in \mathbb{R}^K$ is a K-dimensional feature vector for pixel i, computed from data via a shallow CNN [13]. $\mathbf{M} \succeq 0 \in \mathbb{R}^{K\times K}$ is a PSD *metric matrix*, also learned from data, and $d^* \gg 0$ is a parameter. By (13), larger feature distance means smaller edge weights, and thus $\mathcal{G}^d$ is a *similarity* graph. Signed edges are important, so that interpolation weights in $\mathbf{\Theta}$ can be adjusted in both directions.

3.1.2. Cascading Filter Interpretation

Towards an intuitive interpretation, we rewrite (12) as

$$\mathbf{x}_N^* = \underbrace{(\mathbf{I}_M + \mathbf{\Theta}\mathbf{P}_{M,N})^{-1}}_{\mathbf{\Theta}_P}\mathbf{\Theta}\,\mathbf{y} = \mathbf{\Theta}_P\,\mathbf{\Theta}\,\mathbf{y}. \tag{14}$$

We see that using perturbation matrix $\mathbf{P}_{M,N}$ to augment adjacency matrix $\mathbf{A}_{M,N}$ is equivalent to filtering the original interpolated output $\mathbf{\Theta}\mathbf{y}$ *again* using a new filter $\mathbf{\Theta}_P$. In one special case when $\mathbf{P}_{M,N} = \mathbf{0}_{M,N}$, $\mathbf{\Theta}_P = \mathbf{I}_M$, as expected.

3.1.3. Multiple Perturbations via Cascades

Can we leverage the cascading filter interpretation in (14) to implement multiple perturbations? Considering $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{P}$ as the original adjacency matrix with corresponding filter $\tilde{\mathbf{\Theta}} = \mathbf{\Theta}_P\mathbf{\Theta}$, we can perturb it again with $\mathbf{P}^{(2)}$, resulting in $\tilde{\mathbf{A}}^{(2)} = \tilde{\mathbf{A}} + \mathbf{P}^{(2)} = \mathbf{A} + \mathbf{P} + \mathbf{P}^{(2)}$. The interpolated signal $\mathbf{x}_N^{(2)}$ is then

$$\begin{aligned}\mathbf{x}_N^{(2)} &= (\mathbf{I}_M + \tilde{\mathbf{\Theta}}\mathbf{P}_{M,N}^{(2)})^{-1}\tilde{\mathbf{\Theta}}\,\mathbf{y}\\ (\mathbf{I}_M + \mathbf{\Theta}_P\mathbf{\Theta}\mathbf{P}_{M,N}^{(2)})\mathbf{x}_N^{(2)} &= \mathbf{x}_N^* = \mathbf{\Theta}_P\,\mathbf{\Theta}\,\mathbf{y}.\end{aligned} \tag{15}$$

However, to compute linear system (15), one must first compute matrix inverse $\mathbf{\Theta}_P \triangleq (\mathbf{I}_M + \mathbf{\Theta}\mathbf{P}_{M,N})^{-1}$, which is expensive. We seek a more computation-efficient alternative in the sequel.

3.2. Second Perturbation Matrix via Denoising

Revisiting the original MAP optimization (4), it is apparent that the objective is a combination of two ℓ_2-norm terms:

$$\begin{aligned}&\min_{\mathbf{x}}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \mu\|\mathbf{H}(\mathbf{I} - \mathbf{A})\mathbf{x}\|_2^2\\ &= \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \mu\mathbf{x}^\top\underbrace{(\mathbf{I} - \mathbf{A})^\top\mathbf{H}^\top\mathbf{H}(\mathbf{I} - \mathbf{A})}_{\mathbf{B}}\mathbf{x}\end{aligned} \tag{16}$$

Thus, perturbing the symmetric and PSD matrix $\mathbf{B}$ with another symmetric and PSD Laplacian matrix $\mathbf{P}^{(2)} \in \mathbb{R}^{N\times N}$, corresponding to an *undirected* positive graph $\mathcal{G}^u$ inter-connecting *only* the N interpolated pixels for denoising, results in

$$= \underbrace{\|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \mu\,\mathbf{x}^\top\mathbf{B}\mathbf{x}}_{h(\mathbf{x})} + \underbrace{\mu\,\mathbf{x}^\top\mathbf{G}^\top\mathbf{P}^{(2)}\mathbf{G}\mathbf{x}}_{g(\mathbf{x})}. \tag{17}$$

where $\mathbf{G} = [\mathbf{0}_{N,M}\ \mathbf{I}_N] \in \mathbb{R}^{N\times(M+N)}$, a complementary matrix to $\mathbf{H}$, selects only the N interpolated pixels from $\mathbf{x}$. The objective (17) is a sum of two functions: i) $h(\mathbf{x})$ composed of the fidelity term and a prior involving $\tilde{\mathbf{A}}$, and ii) $g(\mathbf{x})$ that is a prior involving $\mathbf{P}^{(2)}$.

To minimize (17) while reusing the earlier mathematical development, we first recall that one expression of the *Douglas-Rachford* (DR) splitting method (eq. (10) to (12) in [17]) is

$$\mathbf{z}(k) = \text{prox}_{\gamma h}(\mathbf{x}(k)) \tag{18}$$

$$\mathbf{v}(k) = \text{prox}_{\gamma g}(2\mathbf{z}(k) - \mathbf{x}(k)) \tag{19}$$

$$\mathbf{x}(k+1) = \mathbf{x}(k) + 2\gamma(\mathbf{v}(k) - \mathbf{z}(k)), \tag{20}$$

where $\gamma \in (0, 1)$ is a DR parameter. We design an iterative optimization method based on DR to compute proximal mappings of $h(\mathbf{x})$ and $g(\mathbf{x})$ alternately until convergence:

h-step: We compute the proximal mapping of $h(\cdot)$ with argument $\mathbf{x}_N(k)$ for $\mathbf{z}_N(k)$ at iteration k:

Table 1: Interpolation performance for different models.

Method	Params	Kodak[20] PSNR | SSIM	Urban100[21] PSNR | SSIM	McM[22] PSNR | SSIM
Bicubic	–	32.59 | 0.9332	28.38 | 0.9176	34.88 | 0.9578
SwinIR [2]	3,133,969	35.79 | 0.9612	31.52 | 0.9519	39.75 | 0.9805
Restormer [3]	25,439,542	36.26 | 0.9617	32.02 | 0.9536	39.86 | 0.9792
uGTV [13]	319,115	36.81 | 0.9653	32.92 | 0.9610	40.43 | 0.9818
Ours	**262,585**	**37.04** | **0.9671**	**33.85** | **0.9657**	**41.04** | **0.9829**

$$(\mathbf{I}_N + \mathbf{\Theta}\mathbf{P}_{M,N})\mathbf{z}_N(k) = \mathbf{\Theta}(\mathbf{y} + \frac{1}{2\gamma}(\mathbf{x}_N(k-1) - \mathbf{x}_N(k))). \tag{21}$$

Note that we can compute $\mathbf{z}_N(k)$ in (21) via BiCG without inverting $\mathbf{\Theta}$, similar to (12). See Appendix A for a derivation.

g-step: Given $\mathbf{z}_N(k)$, we compute the proximal mapping of $g(\cdot)$ with argument $2\mathbf{z}_N(k) - \mathbf{x}_N(k)$ for $\mathbf{v}_N(k)$. The solution is

$$(2\mu\mathbf{P}^{(2)} + \gamma^{-1}\mathbf{I}_N)\mathbf{v}_N(k) = \gamma^{-1}\left(2\mathbf{z}_N(k) - \mathbf{x}_N(k)\right). \tag{22}$$

See Appendix B for a derivation.

Given $\mathbf{x}_N(k)$, computed $\mathbf{z}_N(k)$ and $\mathbf{v}_N(k)$, $\mathbf{x}_N(k+1)$ for iteration $k+1$ is updated using (20). h-step, g-step and $\mathbf{x}$-update are repeated until convergence.

We unroll DR iterations into neural layers to compose a feed-forward network, so that parameters can be optimized using data; see Fig. 1 for an illustration. Specifically, $\mathbf{P}^{(2)}$ specifies an N-node undirected graph $\mathcal{G}^u$, where *positive* edge weights $w'_{i,j}$ is computed as a function of feature distance $d'_{i,j}$:

$$w'_{i,j} = \exp\left(-d'_{i,j}\right), \qquad d'_{i,j} = \mathbf{f}_j'^\top\mathbf{R}\mathbf{f}'_i. \tag{23}$$

$\mathbf{f}' \in \mathbb{R}^K$ is a feature vector for pixel i, which is also computed from data via a shallow CNN. $\mathbf{R} \succeq 0 \in \mathbb{R}^{K\times K}$ is a PSD metric matrix, also learned from data. This ensures $\mathbf{P}^{(2)}$ is PSD.

4. EXPERIMENTS

4.1. Experimental Setup

All experiments were carried out in a Python 3.12 environment, with all models implemented using PyTorch and trained on NVIDIA GeForce RTX 3090 hardware. To train each learning model, we employed the DIV2K[23] dataset, which contains 800 high-resolution training images and 100 validation images. Due to the high image resolution, we randomly segmented each image into 64×64 size patches for model training. For model evaluation, we employed the Kodak [20], Urban100 [21], and McM [22] datasets. Owing to the model's architectural design, our method does not operate directly on the entire image. Instead, images are divided into 64×64 size patches for interpolation. Overlapping regions and edge blurring are weighted to achieve smooth fusion for image reconstruction. In contrast, the comparison method performs interpolation directly on the full image.

We focused on single-channel Y (luminance) image interpolation applications. To create input images, we started with a full Y-channel image and then removed approximately 50% of pixel values in a checkerboard pattern, equivalent to retaining two out of every four pixels in a checkerboard distribution. The model architecture

Table 2: Quantitative results of different configurations.

Configuration	Θ Type	Matrix P	P Weights	Matrix $P^{(2)}$	PSNR (dB)	ΔPSNR (dB)
Baseline	Fixed	×	–	×	31.45	–
Variant 1	Learnable	×	–	×	32.48	+1.03
Variant 2	Learnable	✓	Positive	×	30.85	-0.60
Variant 3	Learnable	✓	Signed	×	34.60	+3.15
Proposed	**Learnable**	✓	Signed	✓	**35.37**	**+3.92**

employs a two-stage cascaded perturbation network ($\mathbf{P} + \mathbf{P}^{(2)}$), featuring a shallow CNN as the feature extractor with 4 convolutional layers, each producing 48 feature maps. The number of unfolded CG layers is set to 15, and the number of unfolded DR layers is also set to 15. The batch size is 8, the learning rate is 1e-3, and an early stopping strategy with patience=5 is used, employing the Adam optimizer.

To comprehensively evaluate the performance of the proposed two-stage cascaded restoration network, we considered four representative baseline methods for comparison: traditional interpolation methods (Bicubic interpolation), CNN-based methods (SwinIR [2]), Transformer-based methods (Restormer [3]), and graph signal processing-based methods (uGTV [13]). These baselines encompass the mainstream technical approaches in the current image restoration field. We evaluate performance on test images using two common image metrics: Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM)[24].

4.2. Experimental Results

Table 1 shows the interpolation performance of different models. We observe that our model has the smallest number of parameters and achieves the best overall performance across all three test sets. Although uGTV is also based on GSP and employs an unroll architecture, our method achieves superior PSNR and SSIM performance with fewer parameters while better capturing high-frequency details. Leveraging a "dual-driven by model and data" design, the proposed method demonstrates more prominent advantages in efficiency, accuracy, and detail restoration.

To evaluate the contributions of each component in the proposed framework, we conducted comprehensive ablation studies. Table 2 presents quantitative comparisons of different architectural configurations on 100 validation images from the DIV2K dataset.

Through in-depth analysis of ablation experiment results, we draw the following key conclusions: The learnable $\mathbf{\Theta}$ matrix achieves a 1.03 dB improvement in PSNR (Variant 1), demonstrating its superior adaptability to image feature distributions. The first perturbation matrix $\mathbf{P}$, constructed using the positive-edge method, results in a performance drop of 0.60 dB (Variant 2), indicating limitations in optimization with unsigned constraints. In contrast, the signed-edge constructed matrix $\mathbf{P}$ using (13) contributes a 3.15 dB gain (Variant 3), demonstrating the effectiveness of signed feature optimization. The complete model (learnable $\mathbf{\Theta} + \mathbf{P} + \mathbf{P}^{(2)}$) achieves 35.19 dB, representing a significant 3.74 dB improvement over the baseline, fully validating the synergistic optimization effect of the dual-perturbation modules. The results demonstrate that the learnable parameters and dual-perturbation cascade design employed in this paper effectively enhance image reconstruction quality, with distinct synergistic enhancement effects among the components.

The visual comparison in Fig. 2 reveals that the bicubic interpolation method exhibits noticeable blurring and detail loss in the restored results, particularly in areas with complex textures. Deep learning-based methods like SwinIR and Restormer improve image clarity to some extent but still suffer from edge texture discontinuities

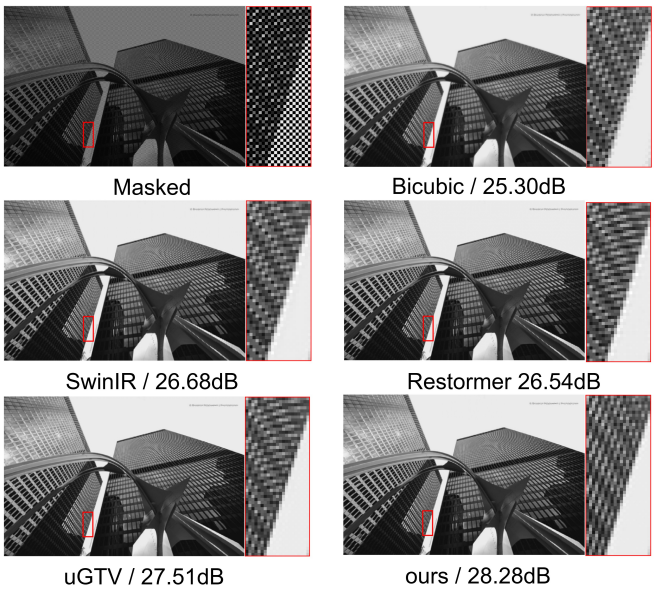

Fig. 2: Visual interpolation results for the image "Urban062".

or excessive smoothing. The uGTV method demonstrates certain advantages in structural preservation but falls short in restoring high-frequency details. In contrast, the proposed method more effectively captures and reconstructs high-frequency information, significantly enhancing local texture sharpness and naturalness while preserving overall structural integrity.

In summary, the proposed two-stage cascaded restoration network demonstrates significant advantages in both quantitative metrics and subjective visual quality. First, the model excels at capturing high-frequency information in images, outperforming existing methods in texture detail and edge restoration. Second, its "dual-driven by data and model" design incorporates an unroll-based DR layer structure, substantially reducing parameter complexity while fully leveraging data characteristics. This achieves both lightweight implementation and high precision. These dual advantages ensure a balanced performance-efficiency tradeoff, providing strong feasibility for practical applications.

5. CONCLUSION

Leveraging a recent interpolator theorem [15] that maps a linear interpolator $\mathbf{\Theta}$ to a directed graph filter that is a solution to a MAP problem using the graph shift variation (GSV) [16] as signal prior, we build a lightweight and interpretable neural net by unrolling an iterative algorithm solving the MAP problem. The key novelty is that we can initialize the network as $\mathbf{\Theta}$, thus guaranteeing a baseline performance level, before introducing two perturbation matrices corresponding to directed and undirected graphs for further gain, implemented efficiently via Douglas-Rachford iterations [17]. Experiments demonstrate SOTA image interpolation results with a drastic reduction in network parameters.

A. PROOF OF (21)

The proximal mapping for $h(\mathbf{x}(k))$ is

$$\min_{\mathbf{x}} \|\mathbf{y}-\mathbf{H}\mathbf{x}\|_2^2 + \mu\,\mathbf{x}^\top\mathbf{B}\mathbf{x} + \frac{1}{2\gamma}\|\mathbf{x}-\mathbf{x}(k)\|_2^2 \tag{24}$$

Taking the derivative w.r.t. $\mathbf{x}$ and set it to zero, we get

$$\begin{aligned}(2\mathbf{H}^\top\mathbf{H} + 2\mu\mathbf{B} + \gamma^{-1}\mathbf{I})\mathbf{x}^* &= 2\mathbf{H}^\top\mathbf{y} + \gamma^{-1}\mathbf{x}(k)\\ \underbrace{(\mathbf{H}^\top\mathbf{H} + \mu\mathbf{B})}_{\mathbf{C}}\mathbf{x}^* &= \mathbf{H}^\top\mathbf{y} + \frac{1}{2\gamma}(\mathbf{x}(k)-\mathbf{x}^*)\end{aligned} \tag{25}$$

Assuming $\mathbf{x}(k)\to\mathbf{x}^*$, we approximate $\mathbf{x}(k)-\mathbf{x}^*$ as $\mathbf{x}(k-1)-\mathbf{x}(k)$:

$$\begin{aligned}\mathbf{C}\mathbf{x}^* &= \mathbf{H}^\top\mathbf{y} + \frac{1}{2\gamma}(\mathbf{x}(k-1)-\mathbf{x}(k))\\ (\mathbf{A}_{M,N}+\mathbf{P}_{M,N})\mathbf{x}_N^* &= \mathbf{y} + \frac{1}{2\gamma}(\mathbf{x}_N(k-1)-\mathbf{x}_N(k))\\ (\mathbf{I}_N + \mathbf{\Theta}\mathbf{P}_{M,N})\mathbf{x}_N^* &= \mathbf{\Theta}(\mathbf{y} + \frac{1}{2\gamma}(\mathbf{x}_N(k-1)-\mathbf{x}_N(k))\end{aligned} \tag{26}$$

where the last two lines follow the same derivation as (12) .

B. PROOF OF (22)

Denote by $\mathbf{u}(k) \triangleq 2\mathbf{z}(k) - \mathbf{x}(k)$. By definition of $g(\cdot)$, $\text{prox}_{\gamma g}(\mathbf{u}(k))$ is the argument that minimizes

$$\min_{\mathbf{x}}\ \mu\mathbf{x}^\top\mathbf{G}^\top\mathbf{P}^{(2)}\mathbf{G}\mathbf{x} + \frac{1}{2\gamma}\|\mathbf{x}-\mathbf{u}(k)\|_2^2. \tag{27}$$

We take the derivative w.r.t. $\mathbf{x}$ and set it to zero:

$$\begin{aligned}2\mu\mathbf{G}^\top\mathbf{P}^{(2)}\mathbf{G}\mathbf{x}^* - \gamma^{-1}\mathbf{u}(k) + \gamma^{-1}\mathbf{x}^* &= 0\\ (2\mu\mathbf{P}^{(2)} + \gamma^{-1}\mathbf{I}_N)\mathbf{x}_N^* &= \gamma^{-1}\mathbf{u}_N(k).\end{aligned} \tag{28}$$

C. REFERENCES

[1] J. R Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," Tech. Rep., USA, 1994.

[2] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte, "Swinir: Image restoration using swin transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, October 2021, pp. 1833–1844.

[3] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang, "Restormer: Efficient transformer for high-resolution image restoration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5728–5739.

[4] Alex Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[6] Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank Reddi, Sanjiv Kumar, and Suvrit Sra, "Why are adaptive methods good for attention models?," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, Eds. 2020, vol. 33, pp. 15383–15393, Curran Associates, Inc.

[7] Vishal Monga, Yuelong Li, and Yonina C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Processing Magazine*, vol. 38, no. 2, pp. 18–44, 2021.

[8] Yaodong Yu, Sam Buchanan, Druv Pai, Tianzhe Chu, Ziyang Wu, Shengbang Tong, Benjamin Haeffele, and Yi Ma, "White-box transformers via sparse rate reduction," in *Advances in Neural Information Processing Systems*, A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds. 2023, vol. 36, pp. 9422–9457, Curran Associates, Inc.

[9] A. Ortega, P. Frossard, J. Kovacevic, J. M. F. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," in *Proceedings of the IEEE*, May 2018, vol. 106, no.5, pp. 808–828.

[10] G. Cheung, E. Magli, Y. Tanaka, and M. Ng, "Graph spectral image processing," in *Proceedings of the IEEE*, May 2018, vol. 106, no.5, pp. 907–930.

[11] Jiahao Pang and Gene Cheung, "Graph Laplacian regularization for image denoising: Analysis in the continuous domain," *IEEE Transactions on Image Processing*, vol. 26, no. 4, pp. 1770–1785, 2017.

[12] Y. Bai, G. Cheung, X. Liu, and W. Gao, "Graph-based blind image deblurring from a single photograph," *IEEE Transactions on Image Processing*, vol. 28, no.3, pp. 1404–1418, 2019.

[13] VIET HO TAM THUC DO, Parham Eftekhar, Seyed Alireza Hosseini, Gene Cheung, and Philip Chou, "Interpretable lightweight transformer via unrolling of learned graph smoothness priors," in *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, Eds. 2024, vol. 37, pp. 6393–6416, Curran Associates, Inc.

[14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2014.

[15] Niruhan Viswarupan, Gene Cheung, Fengbo Lan, and Michael S. Brown, "Mixed graph signal analysis of joint image denoising / interpolation," in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024, pp. 9431–9435.

[16] Michael Elad Yaniv Romano and Peyman Milanfar, "The little engine that could: Regularization by denoising (red)," in *SIAM Journal on Imaging Sciences*, 2017, vol. 10, no.4, pp. 1804–1844.

[17] Pontus Giselsson and Stephen Boyd, "Linear convergence and metric selection for Douglas-Rachford splitting and ADMM," *IEEE Transactions on Automatic Control*, vol. 62, no. 2, pp. 532–544, 2017.

[18] Henk A. Van Der Vorst, "Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems," *SIAM Journal on Scientific and Statistical Computing*, vol. 13, pp. 631, 1992.

[19] Magnus R Hestenes and E. L. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards (United States)*, vol. 49, 1952.

[20] Eastman Kodak, "Kodak lossless true color image suite (photocd pcd0992)," *URL http://r0k. us/graphics/kodak*, vol. 6, pp. 2, 1993.

[21] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5197–5206.

[22] Lei Zhang, Xiaolin Wu, Antoni Buades, and Xin Li, "Color demosaicking by local directional interpolation and nonlocal adaptive thresholding," *Journal of Electronic imaging*, vol. 20, no. 2, pp. 023016–023016, 2011.

[23] Eirikur Agustsson and Radu Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017, pp. 126–135.

[24] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.