SINGLE IMAGE INTERPOLATION EXPLOITING SEMI-LOCAL SIMILARITY

Lantao Yu, Michael T. Orchard

Rice University
Department of Electrical and Computer Engineering
Houston, TX, 77005, USA

ABSTRACT

This paper explores the modeling and exploitation of semi-local similarity in natural images to address the ill-posed nature of image interpolation. Our approach distinguishes itself from prior approaches by direct and careful use of semi-local similar patches to interpolate each individual patch. Our work uses a simple, parallelizable algorithm without the need to solve complicated optimization problems. Experimental results demonstrate that our interpolated images achieve significantly higher objective and subjective quality compared with those from state-of-the-art algorithms.

Index Terms— image interpolation, semi-local similarity, non-local similarity, image modeling

1. INTRODUCTION

Image interpolation addresses the problem of generating a high-resolution image from its low-resolution counterpart. Interpolation algorithms can be understood as reflecting some model for the relationship between the high-resolution and low-resolution images, and the model's accuracy and generality often dictate algorithm performance.

Early interpolation algorithms (e.g. bicubic) employed linear, spatially invariant models that failed to capture the fast evolving properties of edges and that generated interpolated images with blurry edges, significant artifacts and low visual quality. Subsequent contributions employed spatially varying models adaptive to local structures, with improved results. For example, NEDI [1] modeled orientation-invariance between the low- and high-resolution covariance structure in the vicinity of edges to improve interpolation of local oriented structures. Later algorithms [2, 3, 4, 5, 6] refined the directional modeling of NEDI [1], reflecting various priors on natural images, with improved performance. SME [7] used cubic-spline directional interpolators matching the local orientations to exploit smoothness along edge contour directions. It achieves similar performance as [2, 3, 4, 5, 6].

Recent image interpolation algorithms [8, 9] have been based on modeling patches of images as linear combinations of a sparse dictionary of patches (atoms). Started with an interpolated image via a simple, spatially-invariant operator (e.g. cubic spline interpolator), the high-resolution image is iteratively updated via linearly combining the iteratively generated atoms based on "similar" patches collected from throughout the image (non-local similarity). Impressive results have been demonstrated based on this approach.

This paper adopts a variation of a non-local similarity model (we refer to it as "semi-local similarity" [1]) in natural images: namely, it is often true that a given patch can be well approximated by some linear combination of "similar" patches collected within its neighborhood in the image. For the case of 2-by-2 interpolation, we consider each patch of the high-resolution image to comprise 4 subsampling phases, with one of those phases containing the measured pixel values from the low-resolution image, and the others representing missing pixels to be estimated. When applying our semi-local similarity model to estimate "missing pixel" phases of a given high-resolution patch, we recognize that some of the "similar" patches that are linearly combined in the model contain measured pixel values in positions corresponding to the missing pixels to be estimated. We define an iterative approach for updating "missing pixel" phases of patches from information drawn from "measured pixel" phases of similar patches, which converges to a reliable estimate of the high-resolution image. The approach of [10] takes a similar approach and achieves high performance.

This paper proposes an iterative algorithm to exploit semi-local similarity in single image interpolation task. This algorithm progressively updates an interpolated image starting with a tentatively interpolated image. Within each iteration, the image is first decomposed into overlapping patches. Then for each individual patch, multiple similar patches has been searched whose pixel values are linearly combined to update "missing pixel" phases of this patch. Each iteration ends up with synthesizing all the updated patches into an interpolated image. An interpolated image is generated after a few iterations. Our work is different from [10] in a sense that we have a careful management of this iterative scheme aiming at best exploiting semi-local similarity. Our algorithm is not only simple compared with [8, 9] in a sense, without the need

[1] Semi-local similarity is usually treated as a practical version of non-local similarity by specifying searching similar patches within a neighborhood to reduce the computational cost. People commonly introduce their work as using "non-local similarity", while practically use "semi-local similarity".

to address complicated optimization problem, but also parallelizable since updating each patch only depends on its semi-local patches. This paper presents our algorithm and demonstrates its effectiveness in the task of interpolating an image by a factor of 2 and 3, both horizontally and vertically. Experimental results show our algorithm remarkably enhances image quality compared with current state-of-the-art algorithms in PSNR and SSIM senses.

The paper is organized as follows: Section 2 introduces the scheme of our interpolation algorithm. Section 3 demonstrates the experimental results applying our algorithm on standard test images. Section 4 compares our algorithms with related work. We conclude the contribution of the paper and highlight the future research directions in Section 5.

2. INTERPOLATION SCHEME

Without the loss of generality, we assume that the low-resolution imageI_L of size $M \times N$ comes from directly downsampling a high-resolution image I_H of size $2M \times 2N$, i.e. $I_L(p,q) = I_H(2p-1, 2q-1), 1 \le p \le M, 1 \le q \le N$. We wish to find $\hat{I}_H$, so that $\hat{I}_H$ is as close to I_H as possible.

For notational convenience, we denote the grid containing measured pixel values as measurement grid, or OO grid (shown as black solid squares in Figure 1(a)); the grid of coordinates whose vertical and horizontal coordinates are all even as EE grid; the grid of coordinates whose vertical coordinates are even while horizontal coordinates are odd as EO grid; the grid of coordinates whose vertical coordinates are odd while horizontal coordinates are even as OE grid. For patches whose upper-left coordinates are at OO, OE, EO, EE grid as P_{oo}, P_{oe}, P_{eo}, P_{ee} type, respectively. Within each patch, we label the pixels at odd vertical and odd horizontal coordinates as OO-position-phase pixels, shown as the black solid squares in Figure 1 (b); the pixels at odd vertical and even horizontal coordinates as OE-position-phase pixels, shown as the blue solid squares in Figure 1 (b); the pixels at even vertical and odd horizontal coordinates as EO-position-phase pixels, shown as the red solid squares in Figure 1 (b); the pixels at even vertical and even horizontal coordinates as EE-position-phase pixels, shown as the green solid squares in Figure 1 (b).

This section proposes a scheme that exploits the pixels on OO grid, i.e. I_L, to estimate the pixel values on OE, EO, EE grids based on semi-local similarity, thereby generating $\hat{I}_H$.

2.1. Basic Idea

Based on semi-local similarity, in I_H, each individual patch has its similar patches within its own semi-local region. (We treat the patch as P_{oo} type and label it as P_i, with the size of $n \times n$, where n is even). P_i's similar patches are likely to be categorized as P_{oo}, P_{oe}, P_{eo} and P_{ee} type, as opposed to only as P_{oo} type. Within the pool of similar patches, P_{oe}, P_{eo} and P_{ee} type of patches will have OE-, EO- and EE-position-phase pixels, respectively, all at OO grid. We will operate on these known, ground-truth pixels to update corresponding position-phase pixels in P_i.

To update each type of position-phase pixels in P_i, we linearly combine the same type of position-phase pixels in corresponding type of P_i's similar patches. An interpolated image is then updated by synthesizing all updated P_is. It is worth mentioning that although I_H is not available in actual interpolation process, we treat an intermediately interpolated image as I_H and assume semi-local similarity still holds.

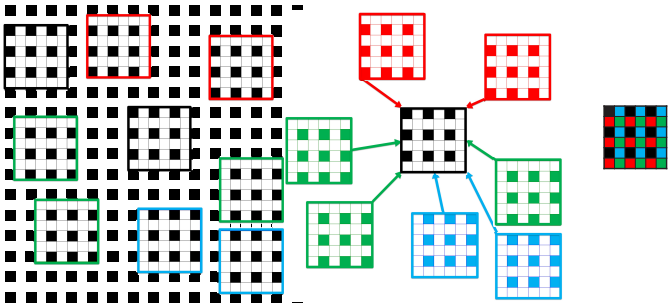


Fig. 1. An illustration of the filtering scheme for each patch. Given a patch P_i whose upper-left pixel is at OO grid (surrounded by a black frame), similar patches whose upper-left pixels at OE, EO, EE grid are found (a). Each of these similar patches has pixels on OO grid that well approximate corresponding pixels off the OO grid in P_i (b). By replacing these pixels in P_i with the filtered corresponding measured pixels in similar patches, an interpolated patch is generated (c).

2.2. Keystones

Our interpolation's performance is dictated by how to find P_i's similar patches and how to compute these patches' weights. Due to page limitation, we focus on introducing the keystones of our interpolation scheme in this subsection.

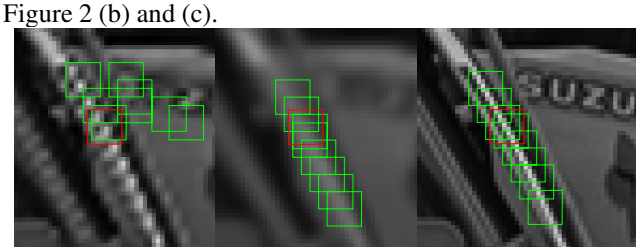
P_i's non-local similar patches are searched in a square window centered at P_i's upper-left pixel coordinate. We use metrics to quantify the similarity between P_i and any patch in the search window in the range of $[0, 1]$ and ensure that the more similar the patch pair, the larger the similarity value. Given the positions of the identified similar patches, the weights, i.e. a $K \times 1$ vector $\boldsymbol{\omega}$, are computed by minimizing the regularized error of approximating K patches to P_i. To offer a closed-form solution of $\boldsymbol{\omega}$ and penalize the weights of less similar patches, we use Generalized Ridge Regression [11] to compute $\boldsymbol{\omega}$ which takes the form of Equation 1:

$$\boldsymbol{\omega} = \arg\min \left[(\boldsymbol{Q}\boldsymbol{\omega} - \boldsymbol{p})^T (\boldsymbol{Q}\boldsymbol{\omega} - \boldsymbol{p}) + \lambda \boldsymbol{\omega}^T \boldsymbol{S} \boldsymbol{\omega} \right], \quad (1)$$

where $\boldsymbol{S}$ is a $K \times K$ diagonal matrix whose diagonal elements are similarity measures; $\boldsymbol{p}$ stores P_i's pixel values to be approximated by corresponding similar patches whose pixel values stored in $\boldsymbol{Q}$; both the size of $\boldsymbol{p}$ and $\boldsymbol{Q}$ may vary by stages of the scheme; λ controls the regularization strength.

Note that the correctness of the coordinates of similar patches plays a crucial role in $\hat{I}_H$'s quality, since it determines the rough structures of $\hat{I}_H$ such as the orientation of

edges. Unfortunately, the starting tentative image in the first iteration often misleads the identification of similar patches. When a bicubic-interpolated or lanczos-interpolated image from I_L, we call I_H^0, is treated as the starting tentative image, shown as Figure 2 (a), due to the strong aliasing around edges, the choice of a near-edge patch's similar patches are often not reflecting the local structures (shown as Figure 2 (a)), which in turn lead to artifacts in $\hat{I}_H$, a typical example of which can be found in Figure 3 (h). To more reliably find similar patches, we remove aliasing in I_L using a low-pass filter to ensure local structures are identifiable in filtered I_L. A high-resolution image I_H^{lp} is then bicubic-interpolated from filtered I_L to find similar patches in the first iteration. The similar patches identified from I_H^{lp} form consistent orientation with those identified from the ground-truth image, shown in Figure 2 (b) and (c).



(a) (b) (c)

Fig. 2. An illustration of how aliasing influences the choice of similar patches and how low-pass filtering addresses this problem. Given P_i, shown as the red block, top-8 P_{ee}-type similar patches shown as the green blocks are searched near P_i by operating on I_H^0 (a), I_H^{lp} (b), and I_H (c), respectively.

It is also worth noticing that in the aforementioned scheme, the pixels off OO grid are not treated as the bases to approximate P_i due to their limited correctness. However, after a few iterations, these pixels are so close to ground-truth that they are eligible of updating P_i. We thus use all the pixels in P_i and P_i's similar patches to update P_i via Equation 1.

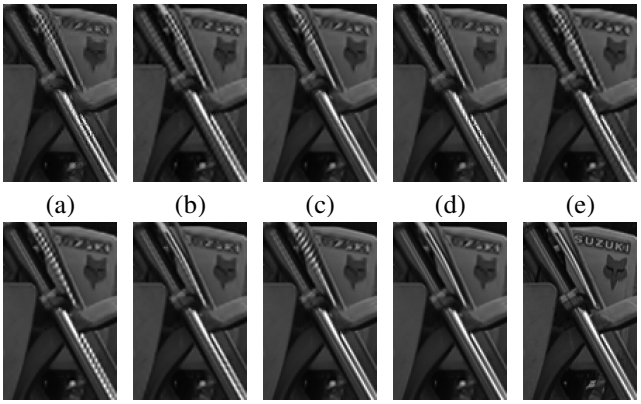
STAGE 1: Preprocess I_L to obtain a less-aliased tentatively image to find reliable filter supports and weights.

STAGE 2 and STAGE 3: Exploit pixels at OO grid. The major difference between STAGE 2 and STAGE 3 is that STAGE 2 computes the filter weights by approximating only the measured pixels in P_i, while STAGE 3 computes them by approximating all the pixels in P_i.

STAGE 4: Exploit the pixels both on and off OO grid. The difference between STAGE 1–3 and STAGE 4 is that when updating P_i, STAGE 4 computes and apply weights to all the pixels on the grid rather than the pixels only at OO grid.

3. EXPERIMENTS AND RESULTS

Our algorithm operates on 23 common test images, some of which can be found in the USC-SIPI image database and the Berkeley Segmentation Database[2], via interpolating the downsampled-by-2 versions of them by a factor of 2(X2), and the downsampled-by-3 version of them by a factor of 3(X3), both horizontally and vertically. The parameters involved are appropriately chosen. We compare our X2 results with the results from [1, 7, 2, 3, 6, 8, 9, 10] [3] and our X3 results with the results from [8, 9, 10] in Table 1 and Figure 3. PSNR(Peak Signal-to-Noise Ratio) and SSIM (Structural Similarity [12]) are utilized as image quality metrics.



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Fig. 3. The zoom-in comparison of interpolated edges (X2) in *Motorbike* from different methods with the ground-truth. (a):[1], (b):[7], (c):[2], (d):[3], (e):[6], (f):[8], (g):[9], (h):[10], (i):Ours, (j): Ground-Truth.

Table 1 demonstrates that in X2 scenario, our algorithm achieves the highest PSNR in 21 out of 23 images and the highest SSIM [12] in 20 out of 23 images. Our results have both the highest average PSNR and the highest average SSIM which are 0.32 dB/0.0040 better than ANSM [9], 0.33 dB/0.0038 better than NLPC [10] and 0.44 dB/0.0051 better than NARM [8]; in X3 scenario, our algorithm achieves the highest PSNR in 22 out of 23 images and the highest SSIM [12] in 18 out of 23 images. Our results have both the highest average PSNR and the highest average SSIM which are 0.32 dB/0.0069 better than ANSM [9], 0.22 dB/0.0047 better than NLPC [10] and 0.57 dB/0.0128 better than NARM [8].

Our algorithm also interpolates images with high visual quality as illustrated in Figure 3. We can tell that our algorithm clearly reconstructs all the edges along upper-left to bottom-right direction, preserving both smoothness along the contour direction and sharpness across the profile direction, without any noticeable artifacts.

Table 2. Computational Cost Comparison (in seconds) between Ours and [8, 9] in a Scenario of Interpolating a 128×128 Image to a 256×256 Image.

ALGORITHM & COMPUTING PLATFORM	SECONDS
[8] (MATLAB, NON-PARALLEL)	55.5
[9] (MATLAB, C/C++ MEX)	1265.0
Ours (MATLAB, NON-PARALLEL)	155.6
Ours (MATLAB, PARALLEL)	13.3
Ours (C++, PARALLEL)	3.6

We test different algorithms' runtimes of interpolating a 128×128 image to a 256×256 image on a 2.6G Hz 18-Core

[2]For color images, only luminance channel will our algorithm be applied.

[3]We thank Professor Shuyuan Zhu at UESTC for providing the results of his algorithm [6], thank Professor Xianming Liu at Harbin Institute of Technology for providing the Matlab source code of his algorithm [3].

Table 1. Comparison of PSNRs (in decibels)/SSIMs of the interpolated images between Ours and [1, 7, 2, 3, 6, 8, 9, 10].

	[1] (X2)	[7] (X2)	[2] (X2)	[3] (X2)	[6] (X2)	[8] (X2)	[9] (X2)	[10] (X2)	Ours (X2)	[8] (X3)	[9] (X3)	[10] (X3)	Ours (X3)
Elk	31.47/0.9221	31.49/0.9253	31.85/0.9287	31.53/0.9270	31.72/0.9283	31.95/0.9296	32.51/**0.9326**	32.31/0.9293	**32.72**/0.9314	27.68/0.8477	27.99/0.8569	28.11/0.8551	**28.48/0.8578**
Birds	33.67/0.9493	34.28/0.9501	34.63/0.9504	34.43/0.9527	34.52/0.9523	35.03/0.9538	34.67/0.9525	35.00/0.9542	**35.12/0.9551**	30.01/0.9140	30.40/0.9167	**30.51**/0.9177	30.47/**0.9192**
Butterfly	26.11/0.9222	26.75/0.9287	26.94/0.9377	27.28/0.9399	27.88/0.9436	28.23/0.9475	27.90/0.9444	27.86/0.9441	**28.49/0.9497**	23.46/0.8795	23.54/0.8740	23.54/0.8746	**24.03/0.8880**
Flower	33.21/0.9452	33.75/0.9514	34.09/0.9543	33.88/0.9517	33.67/0.9515	34.41/0.9567	34.13/0.9553	34.22/0.9547	**34.61/0.9577**	30.12/0.8987	30.25/**0.9048**	30.19/0.9006	**30.40**/0.9040
Girl	31.54/0.8620	31.75/0.8689	31.85/0.8685	31.89/0.8686	31.79/0.8682	32.14/0.9567	32.01/0.8717	32.15/0.8721	**32.28/0.8729**	29.13/0.7817	29.01/0.7829	29.25/**0.7866**	**29.31**/0.7844
Hats	32.15/0.9053	32.33/0.9100	32.51/0.9094	32.61/0.9112	32.27/0.9069	32.55/0.9092	32.79/0.9132	32.62/0.9094	**33.21/0.9172**	29.15/0.8377	29.30/0.8421	29.61/0.8459	**29.68/0.8466**
Leaves	26.88/0.9397	27.97/0.9489	28.23/0.9563	28.02/0.9548	28.76/0.9606	29.38/0.9652	28.84/0.9593	29.23/0.9626	**29.95/0.9685**	23.03/0.8733	23.07/0.8719	23.30/0.8718	**23.62/0.8842**
Male	31.64/0.8781	32.16/0.8851	32.16/0.8858	32.20/0.8864	32.20/0.8870	32.41/0.8903	32.40/0.8888	32.50/0.8929	**32.70/0.8950**	28.78/0.7937	28.96/**0.8059**	28.98/0.8031	**29.11**/0.8039
Motorbike	25.99/0.8622	26.71/0.8840	27.00/0.8906	26.63/0.8786	26.62/0.8854	26.85/0.8898	27.09/0.8912	27.16/0.8912	**27.50/0.8979**	22.38/0.7302	22.93/0.7454	23.02/0.7427	**23.46/0.7523**
Boat	29.32/0.8409	29.71/0.8481	29.71/0.8482	29.59/0.8507	29.50/0.8497	29.86/0.8577	30.19/0.8582	30.06/0.8631	**30.26/0.8682**	26.58/0.7647	26.80/0.7665	26.72/0.7650	**26.88/0.7709**
Cameraman	25.68/0.8594	26.24/0.8660	25.99/0.8663	25.90/0.8675	25.65/0.8657	26.04/0.8716	**26.61**/0.8745	26.31/0.8725	26.57/**0.8766**	22.80/0.7817	23.21/0.7874	23.23/0.7888	**23.39/0.7945**
Dragonfly	35.96/0.9623	36.27/0.9658	36.25/0.9638	36.36/0.9653	35.90/0.9612	36.99/0.9671	37.15/**0.9698**	36.91/0.9657	**37.31**/0.9682	32.92/0.9330	33.44/0.9404	33.15/0.9346	**33.75/0.9419**
Fence	21.11/0.7557	23.47/0.7788	22.68/0.7669	22.98/0.7719	23.00/0.7727	23.64/0.7883	23.72/0.7894	23.75/0.7927	**23.91/0.7977**	19.43/0.5974	19.54/0.6059	19.64/**0.6198**	**19.81**/0.6173
Fighter	30.19/0.8409	30.68/0.8498	30.33/0.8408	30.39/0.8495	30.22/0.8408	30.67/0.8603	31.27/0.8634	30.80/0.8612	**31.49/0.8697**	27.43/0.7749	27.62/0.7642	27.83/0.7769	**28.24/0.7876**
Lena	33.95/0.9143	34.64/0.9184	34.74/0.9184	34.51/0.9195	34.68/0.9204	35.09/0.9242	34.87/0.9218	35.08/0.9237	**35.24/0.9253**	31.27/0.8705	31.20/0.8708	31.29/0.8717	**31.56/0.8755**
Peppers	33.26/0.8781	33.37/0.8737	33.45/0.8754	33.73/0.8837	33.53/0.8784	34.07/0.8872	33.83/0.8809	34.14/0.8895	**34.25/0.8922**	31.37/0.8499	31.28/0.8426	31.47/0.8490	**31.66/0.8550**
Sail	32.28/0.9178	32.41/0.9200	32.25/0.9182	32.52/0.9220	32.32/0.9156	32.51/0.9245	32.17/0.9272	32.63/0.9273	**32.98/0.9305**	29.10/0.8622	29.40/0.8646	29.36/0.8729	**29.43/0.8750**
Plane	30.02/0.9110	30.30/0.9142	30.48/0.9156	30.58/0.9177	30.35/0.9159	30.33/0.9193	31.05/0.9223	30.99/0.9221	**31.23/0.9249**	26.75/0.8475	27.37/0.8604	27.46/0.8626	**27.63/0.8656**
Vase	34.51/0.9050	34.43/0.9054	34.66/0.9071	34.81/0.9088	34.45/0.9024	34.89/0.9074	34.92/0.9113	34.93/0.9090	**35.23/0.9132**	32.24/0.8459	32.09/0.8509	32.35/0.8521	**32.46/0.8563**
House	32.14/0.8786	33.19/0.8845	32.87/0.8809	32.99/0.8874	32.76/0.8811	33.49/0.8876	34.46/0.8923	33.93/0.8896	**34.56/0.8961**	29.83/0.8410	30.06/0.8453	30.23/0.8501	**30.30/0.8520**
Parrot	26.45/0.8930	26.85/0.8963	27.34/0.9010	27.16/0.9012	27.34/0.8999	27.19/0.8998	27.45/0.9011	27.58/0.9015	**27.72/0.9020**	23.58/0.8232	24.03/0.8318	24.03/0.8282	**24.25/0.8323**
Texture	20.69/0.8567	21.53/0.8796	21.44/0.8771	21.20/0.8733	21.15/0.8711	21.48/0.8792	22.00/0.8914	21.92/0.8917	**22.13/0.8962**	16.53/0.6659	17.31/0.7061	17.47/0.7200	**17.69/0.7306**
Foreman	36.58/0.9515	36.78/0.9530	37.00/0.9551	37.24/0.9550	37.27/0.9549	**38.41**/0.9562	38.26/**0.9575**	38.03/0.9541	38.36/0.9552	34.16/0.9154	34.51/**0.9283**	34.99/0.9251	**35.05**/0.9275
AVERAGE	30.25/0.8935	30.74/0.9003	30.80/0.9007	30.80/0.9019	30.76/0.9006	31.20/0.9063	31.32/0.9074	31.31/0.9076	**31.64/0.9114**	27.29/0.8230	27.54/0.8289	27.64/0.8311	**27.86/0.8358**

Intel i9 processor using MATLAB (R2018b) and C++ (GCC 7.3.0), which are shown in Table 2. Our algorithm's computational cost (155.6 s), without Parallel Computing Toolbox (PCT)'s optimization, is significantly lower than [9] (1265.0 s) and higher than [8] (55.5 s). Optimized by PCT, the runtime significantly shrinks (13.3 s). Through implementing our algorithm via C++ code supported by Eigen [13] and OpenMP [14], the runtime further shrinks to 3.6 seconds. In conclusion, our algorithm's basic computation is simple, while the number of each basic computation is not small, which jointly give rise to the overall non-trivial complexity. However, since our algorithm is paralleriable, our algorithm can be drastically accelerated optimized by parallel computing.

4. RELATED WORK

Our work and [4, 5, 6, 8, 9, 10] all belong to the type of algorithms that exploit non-local or semi-local similarity. Within this type, we further categorize our approach and [8, 9, 10] as patch-based interpolation approaches which involve both a): the process of computing the weights of semi-local similar patches to approximate a patch, and b): the process of updating the pixels by averaging the contribution of all relevant patches. [4, 5, 6] do not belong to this category since they only satisfy a), but not b). The reason why patch-based algorithms offer more impressive results than [1, 3, 4, 5, 6, 2, 7] is because of the joint effect of a): patch-based regression is less likely to run into the risk of over-fitting the data than pixel-based regression, and b): patch-based algorithms decrease the approximation error via averaging multiple patches' contribution to each pixel when synthesizing patches to an image.

Compared with [8, 9] which are also patch-based approaches, we do not impose sparsity constraint. This offers us abundant bases to approximate each individual patch. Our work is similar to [10] in a way that we are both exploit semi-local similarity as discussed in Section 1. Specifically, our STAGE 2 and 3 bear some resemblance to "Interpolation stage" in [10]. Unfortunately, [10] does not take the ideas of removing aliasing, penalizing the weights of less similar patches and the exploitation of the pixels off measurement grid into consideration.

5. CONCLUSION AND FUTURE WORK

We propose a novel single image interpolation scheme via modeling and exploiting semi-local similarity. We directly use a set of semi-local similar patches to interpolate each individual patch, rooted in a observation that the pixels within an individual patch locating off the measurement grid often have their similar pixels found at the measurement grid within the patch's semi-local similar patches. Our proposed method carefully manages such process and remarkably improves quality as compared to current state-of-the-art methods both in PSNR sense and in SSIM sense.

Although our algorithm has achieved state-of-the-art performance, there exists a large room to improve it. Current definition of "similarity" only takes the perspective of two patches being linearly close. However, "similarity" could also mean the proximity in nonlinear sense. For example, two patches of straight edges of the same sharpness but along distinct orientations are similar yet they have not been exploited in our scheme. Exploiting such nonlinearity offers more abundant bases within the search window so that new similar patches tend to lie within a tighter similarity bound which leads to a better estimation of each individual patch.

Even though "similarity" is all about linear proximity, our work needs to be further improved. For example, using a low-pass filter with a fixed shape to deal with all sorts of local structures with different extents of aliasing in our scheme is suboptimal. Also, using the patches with fixed size is not suitable to exploit semi-local similarity in extremely localized structures and may also result to over-fitting.

In the future, we will explore a boarder definition of "similarity" involving nonlinearity. Also, we will apply adaptive, low-pass filters to the low-resolution image with the awareness of different extent of aliasing in local area to appropriately remove aliasing. Meanwhile, we will use spatially-varying patch size to further exploit semi-local similarity.

6. REFERENCES

[1] Xin Li and Michael T. Orchard, "New edge-directed interpolation," *IEEE transactions on image processing*, vol. 10, no. 10, pp. 1521–1527, 2001.

[2] Xiangjun Zhang and Xiaolin Wu, "Image interpolation by adaptive 2-d autoregressive modeling and soft-decision estimation," *IEEE transactions on image processing*, vol. 17, no. 6, pp. 887–896, 2008.

[3] Xianming Liu, Debin Zhao, Ruiqin Xiong, Siwei Ma, Wen Gao, and Huifang Sun, "Image interpolation via regularized local linear regression," *IEEE Transactions on Image Processing*, vol. 20, no. 12, pp. 3455–3469, 2011.

[4] Kai Guo, Xiaokang Yang, Hongyuan Zha, Weiyao Lin, and Songyu Yu, "Multiscale semilocal interpolation with antialiasing," *IEEE Transactions on Image Processing*, vol. 21, no. 2, pp. 615–625, 2012.

[5] Xianming Liu, Debin Zhao, Jiantao Zhou, Wen Gao, and Huifang Sun, "Image interpolation via graph-based bayesian label propagation," *IEEE Transactions on Image Processing*, vol. 23, no. 3, pp. 1084–1096, 2014.

[6] Shuyuan Zhu, Bing Zeng, Liaoyuan Zeng, and Moncef Gabbouj, "Image interpolation based on non-local geometric similarities and directional gradients," *IEEE Transactions on Multimedia*, vol. 18, no. 9, pp. 1707–1719, 2016.

[7] Stéphane Mallat and Guoshen Yu, "Super-resolution with sparse mixing estimators," *IEEE transactions on image processing*, vol. 19, no. 11, pp. 2889–2900, 2010.

[8] Weisheng Dong, Lei Zhang, Rastislav Lukac, and Guangming Shi, "Sparse representation based image interpolation with nonlocal autoregressive modeling," *IEEE Transactions on Image Processing*, vol. 22, no. 4, pp. 1382–1394, 2013.

[9] Yaniv Romano, M Protter, and M Elad, "Single image interpolation via adaptive non-local sparsity-based modeling," *IEEE Transactions on Image Processing*, 2014.

[10] Dong Sun, Qingwei Gao, and Yixiang Lu, "Image interpolation via collaging its non-local patches," *Digital Signal Processing*, vol. 49, pp. 33–43, 2016.

[11] Arthur E Hoerl and Robert W Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

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

[13] Gaël Guennebaud, Benoît Jacob, et al., "Eigen v3," http://eigen.tuxfamily.org, 2010.

[14] OpenMP Architecture Review Board, "OpenMP application program interface version 3.0," May 2008.