Single image super-resolution by approximated Heaviside functions

Liang-Jian Deng*  Weihong Guo†  Ting-Zhu Huang‡

Abstract

Image super-resolution involves the estimation of a high-resolution image from one or multiple low-resolution images. It is widely used in medical imaging, satellite imaging, target recognition, etc. In this paper, we solve the problem of single image super-resolution from an image intensity function estimation perspective. We assume that the unknown image intensity function is defined on a continuous domain and belongs to a space with a redundant basis. The selection of the redundant basis is based on an observation: an image is composed of smooth and non-smooth components, and we use two classes of approximated Heaviside functions (AHFs) to represent them respectively. The coefficients of the redundant basis are computed iteratively from a given low-resolution image. In addition, we apply the proposed iterative scheme to image patches to reduce computation and storage size. Comparisons with some existing competitive methods show the effectiveness of the proposed method.

Key words: Single image super-resolution, Approximated Heaviside functions, Iterative model

1 Introduction

Image super-resolution (SR) is to estimate a high-resolution (HR) image from one or multiple low-resolution (LR) images. It is called single image super-resolution if there is only one low-resolution input and called multiple image super-resolution if otherwise. As the name suggests, single image super-resolution is more practical when one low-resolution image is available as compared to multiple image super-resolution. Obviously, this is more challenging. In this paper, the proposed method is to deal with single image super-resolution problem. Low-resolution images instead of high-resolution images are sometimes available due to the limitation of hardware devices and high cost. For instance, it is costly and time consuming to obtain high-resolution images by a medical imaging modality Magnetic Resonance Imaging (MRI). Due to long distance and air turbulence, we can not get high-resolution images for synthetic aperture radar (SAR) and satellite imaging. In target recognition, we may not always get high-resolution videos to make accurate recognition. Thus, it is important and useful to develop a more effective image super-resolution algorithm.

There are mainly four categories of image super-resolution methods: interpolation-based methods, learning-based methods, statistics-based methods and other methods such as hybrid method or pixel classification method. In Tab. 1 we briefly list some advantages and disadvantages of methods in each category. Methods in different category are however not completely independent. For instance, some statistics-based methods may involve learning strategies.

Nearest-neighbor interpolation and bicubic interpolation are two classical interpolation methods. Nearest-neighbor interpolation estimates the intensity of an unknown location using that of its nearest neighboring points. However, it often creates a jaggy effect. Bicubic interpolation involves the interpolation of unknown intensities by utilizing a cubic kernel, and may cause a blur effect. More
interpolation methods have been proposed recently [39, 43, 69, 7, 25, 24, 54]. In [24], the authors introduced a contour stencils method to estimate the image contours based on the total variation (TV) along curves. This method can distinguish lines of different orientations, curves, corners, etc., with a computationally efficient formula so that the resulting high-resolution images preserve image details well. In [54], Wang et al. proposed a fast image super-resolution method on the displacement field. This method is based on two phases, one of which uses an interpolation technique to pick up the low-frequency information and the other employs a reconstruction method to recover local high-frequency structures.

Learning-based methods are important tools for many image applications, e.g., image retrieval [63], image reranking [52], and image classification [61]. Recently, many image super-resolution methods are learning-based [23, 22, 17, 50, 28, 21, 46, 34, 41, 76, 10, 60, 59, 68, 75, 60, 10, 67, 38, 8, 51, 52, 15, 27, 38]. Learning-based methods need two large training data sets, one formed by low-resolution images and the other by the corresponding high-resolution images. A relation between the two training data sets is learned and then applied to a given low-resolution image to obtain a high-resolution image. They heavily rely on the selection of training data and involve expensive computation. They are not single image super-resolution approaches in a strict sense as they require extra training data. In particular, most of the learning-based methods are based on an optimization framework that can be used in many image applications, such as image deblurring [73, 74, 12, 14, 40], image decomposition [42], etc. In [59], Yang et al. proposed a sparse signal representation method based on an optimization framework for single image super-resolution. Dong et al. [15] was the first work to apply a deep learning method for single image super-resolution.

Table 1: Four categories for image super-resolution. Since there are no common advantages and disadvantages for “Other methods”, we do not present them in the table.

| Learning-based methods | e.g., Sparse coding [60, 59], deep learning method [15]. More references: [56, 3, 51, 52], etc. | Advantage: accurate. Disadvantage: extra data for training, slow speed. |
| Statistics-based methods | e.g., Maximum a Posterior (MAP), Maximum Likelihood estimator (MLE). | Advantage: accurate. Disadvantage: encounter some artifacts, e.g., blur effect. |
| Other methods | e.g., hybrid method [11], frequency technique [4], etc. More references: [31, 32, 37, 53, 5]. | Advantage: - Disadvantage: - |

Statistics-based methods such as Maximum A Posterior (MAP) and Maximum Likelihood Estimator (MLE) are popular tools for image super-resolution [17, 6, 18, 65]. Fattal [18] utilized statistical edge dependency relating edge features in low and high resolution images to preserve sharp edges. In [17], an alternative approach using $\ell_1$ norm minimization and a bilateral prior based robust regularization was proposed by Farsin et al.

Many other methods have also been proposed for image super-resolution. Examples are hybrid method [11], pixel classification method [12], frequency technique [4], reconstruction method [31, 32, 37, 10, 49] and others [35, 5, 20, 44, 55, 8, 44, 60, 70, 71, 45].

Most existing image super-resolution methods are based on discrete models. In this paper, we propose a continuous model based on approximated Heaviside functions (AHFs) for single image super-resolution. Since an image can be assumed to consist of smooth and non-smooth components such as step edges, we model an image as a sum of two classes of AHFs, one representing the smooth component and the other depicting the non-smooth component. Due to the sparsity of non-smooth
component, we design an $\ell_1$ regularization model and solve it by Alternating Direction Method of Multipliers (ADMM). An iterative model based on the $\ell_1$ model is proposed to preserve more details. We apply the coefficients computed by the iterative model to generate high-resolution images at any upscaling factors. The iterative model is applied to image patches, aiming to get cheap computation and storage size. For images with smooth background, the proposed method may cause ring artifacts and we develop a strategy utilizing image gradient to reduce them. Numerical experiments show that the proposed method is a competitive method when compared with some existing state-of-the-art methods.

To the best of our knowledge, this is the first work to use AHFs on image super-resolution. The proposed method can preserve not only edges, but also the high-frequency details on non-edge regions. The proposed method is completely a single image super-resolution method without the use of training data.

The organization of this paper is as follows. In Section 2, we review Heaviside function and show why we can use it for image super-resolution. In Section 3, we present the proposed method, including the corresponding new model, new algorithm and other strategies. Visual and quantitative experiments are given in Section 4. Finally, we conclude in Section 5.

2 The Heaviside function

The Heaviside function, or the Heaviside step function, is defined as follows (see Fig. 1(a))

$$\phi(x) = \begin{cases} 0, & x < 0, \\ 1, & x \geq 0, \end{cases}$$

which is singular at $x = 0$. In practice, we usually use its approximation, called approximated Heaviside function (AHF), such as $\frac{1}{1+x^2}$ and $\frac{1}{2} + \frac{1}{\pi} \arctan \left( \frac{x}{\xi} \right)$ which approximates to $\phi(x)$ when $\xi \to 0$. In this paper, we use the following approximated Heaviside function,

$$\psi(x) = \frac{1}{2} + \frac{1}{\pi} \arctan \left( \frac{x}{\xi} \right),$$

where $\xi \in \mathbb{R}$ actually controls the smoothness of the approximated Heaviside function. The smaller $\xi$, the sharper the edge (see Fig. 1(b)).

From [33], we know that any function in $\mathcal{L}_p([0, 1]^d)$, $p \in [1, \infty)$, has a best approximation by linear combinations of $m$ characteristic functions of the half-space, where $m$ is any positive integer. Let $H_d$ be a set of functions on $[0, 1]^d$ defined as

$$H_d = \{ f : [0, 1]^d \to \mathcal{R} : f(x) = \psi(v \cdot x + c), v \in \mathcal{R}^d, c \in \mathcal{R} \},$$

where $\psi$ is an approximated Heaviside function. $H_d$ is the set of characteristic function of closed half-spaces of $R^d$.

**Theorem 1** (see [33]) For any positive integer $d$, define $\text{span}_m H_d$ as $\{ \sum_{i=1}^m \omega_i \psi(v_i \cdot x + c_i) \}$, where $\omega_i \in \mathcal{R}$ and $v_i \in \mathcal{R}^d$ and $c_i \in \mathcal{R}$, then it is known that $U_{m \in N^+} \text{span}_m H_d$ is dense in $(\mathcal{L}_p([0, 1]^d), \| \cdot \|_p)$, $p \in [1, \infty)$.

**Theorem 2** (see [33]) For every positive integers $m, d$ and every $p \in [1, \infty)$, $\text{span}_m H_d$ is approximately a compact subset of $(\mathcal{L}_p([0, 1]^d), \| \cdot \|_p)$.

In practical computing, one can only afford to use $\text{span}_m H_d$ for a finite $m$.

In our work, we focus on two-dimensional image super-resolution, so $d = 2$. Here, we assume the underlying image intensity function $f$ is defined on $[0, 1]^2$ and $f \in \mathcal{L}_p([0, 1]^2)$ with $p \in [1, \infty)$, by Theorem 1 and 2, $f$ can be approximated by the following equation,

$$f(z) = \sum_{j=1}^m \omega_j \psi(v_j \cdot z + c_j),$$

where $\omega_j \in \mathcal{R}$, $v_j \in \mathcal{R}^2$ and $c_j \in \mathcal{R}$. We discretize $v_j = \{(\cos \theta_i, \sin \theta_i)', t = 1, 2, \cdots, k\}$ to denote $k$ different directions, and $c_j = \{\frac{1}{q}, \frac{2}{q}, \frac{3}{q}, \cdots, 1\}$ to denote discrete positions, $m = kq$, $z = (x, y)'$. We fix
3 The proposed model and its solution

3.1 The proposed ℓ_1 model and its solution

From Fig. 2 and Fig. 3 it is observed that the corresponding AHFs with bigger ξ represent smooth component well, and the corresponding AHFs with smaller ξ depict non-smooth component such as edges more accurately. Thus, we represent an image with two classes of AHFs, one with a bigger ξ to depict non-smooth component (forming Ψ_1) and the other with a smaller ξ to depict smooth component (forming Ψ_2). The two classes of AHFs form a redundant representation for the underlying intensity function. By this strategy, the vector-form image \( L \) can be approximated by the following discrete formula,

\[
L \approx \Psi_1 \beta_1 + \Psi_2 \beta_2,
\]

where \( \Psi_1, \Psi_2 \in \mathbb{R}^{n \times m} \), and \( \beta_1, \beta_2 \in \mathbb{R}^{m \times 1} \) are representation coefficients. Once the coefficients \( \beta_1, \beta_2 \) are computed, one can apply the computed coefficients to get high-resolution images at finer grids by \( \hat{f} = \Psi_1 \beta_1 + \Psi_2 \beta_2 \), where \( \Psi_1, \Psi_2 \in \mathbb{R}^{N \times m} \) are generated similarly as \( \Psi \) using finer grids. Note that, in our recent work [13], we used the Heaviside function to describe the non-smooth component only, but in this work both smooth and non-smooth components are described using approximated Heaviside functions.

For an image, non-smooth component such as edges are more sparsely distributed than smooth component, we thus enforce ℓ_1 sparsity on \( \beta_2 \) with an ℓ_2 regularity on \( \beta_1 \) accordingly. The optimization model is

\[
\min_{\beta_1, \beta_2} \|L - \Psi_1 \beta_1 - \Psi_2 \beta_2\|_2^2 + \lambda_1 \|\beta_1\|_2^2 + \lambda_2 \|\beta_2\|_1,
\]

where \( \lambda_1, \lambda_2 \) are regularization parameters, \( L \) represents the low-resolution input.
Figure 2: Left panel: surface visualization of $\psi$ under $\xi = 0.1$ and nine random parameter pairs $(\theta, c)$: Right panel: the corresponding 2D intensity images. From left to right and then from top to bottom: $(\frac{4\pi}{5}, \frac{81}{1024})$, $(\frac{4\pi}{5}, \frac{25}{64})$, $(\frac{4\pi}{5}, \frac{175}{256})$, $(\frac{6\pi}{5}, \frac{135}{1024})$, $(\frac{6\pi}{5}, \frac{1}{5})$, $(\frac{8\pi}{5}, \frac{25}{32})$, $(\frac{8\pi}{5}, \frac{5}{64})$, $(\frac{8\pi}{5}, \frac{75}{256})$, $(\frac{8\pi}{5}, \frac{75}{128})$ (for better visualization, we slightly rotate some surfaces).

Figure 3: Left panel: surface visualization of $\psi$ under $\xi = 10^{-4}$ and the same nine parameter pairs $(\theta, c)$ as in Fig. 2 Right panel: the corresponding 2D intensity images. (for better visualization, we slightly rotate some surfaces).
Since $\ell_1$ term is not differentiable, we make a variable substitution for $\beta_2$, and rewrite model (6) as
\[
\min_{\beta_1, \beta_2} \|L - \Psi_1 \beta_1 - \Psi_2 \beta_2\|_2^2 + \lambda_1 \|\beta_1\|_2^2 + \lambda_2 \|u\|_1, \quad \text{s.t.,} \quad u = \beta_2,
\]
where $u$ is the substitution variable. Eq. (7) can be rewritten as
\[
\min_{\beta} \|L - \Psi \beta\|_2^2 + \lambda_1 \|A \beta\|_2^2 + \lambda_2 \|u\|_1, \quad \text{s.t.,} \quad u = B \beta,
\]
where $\Psi = (\Psi_1, \Psi_2)$, $\beta = (\beta_1, \beta_2)'$, $A = (I, 0)$ and $B = (0, I)$. The optimization problem (8) is separable w.r.t $\beta$ and $u$. There are many methods to solve (8). We select the Alternating Direction Method of Multipliers (ADMM) [26, 55, 29].

The augmented Lagrangian of Eq. (8) is
\[
\mathcal{L}(\beta, u, b) = \|L - \Psi \beta\|_2^2 + \lambda_1 \|A \beta\|_2^2 + \lambda_2 \|u\|_1 + \frac{\rho}{2} \|u - B \beta + b\|_2^2,
\]
where $b$ is Lagrangian multiplier with proper size. The problem of minimizing $\mathcal{L}(\beta, u, b)$ is solved by iteratively and alternatively solving the following two subproblems:
\[
\begin{align*}
\beta\text{-subproblem} & : \min_{\beta} \|L - \Psi \beta\|_2^2 + \lambda_1 \|A \beta\|_2^2 + \lambda_2 \|u\|_1 + \frac{\rho}{2} \|u - B \beta + b\|_2^2, \\
u\text{-subproblem} & : \min_{u} \lambda_2 \|u\|_1 + \frac{\rho}{2} \|u - B \beta + b\|_2^2.
\end{align*}
\]

The following ADMM algorithm is used to solve Eq. (8).

**Algorithm 1**

**Input:** Given low-resolution image $L$, $\Psi_1$, $\Psi_2$, $\lambda_1$, $\lambda_2$, $\rho$

**Output:** $\beta$

1. $k \leftarrow 0$, $\beta^{(k)} \leftarrow 0$, $u^{(k)} \leftarrow 0$, $b^{(k)} \leftarrow 0$
2. while not converged do
3. \[k \leftarrow k + 1\]
4. $\beta^{(k)} \leftarrow$ solve subproblem (10) for $u = u^{(k-1)}$, $b = b^{(k-1)}$,
5. $u^{(k)} \leftarrow$ solve subproblem (11) for $\beta = \beta^{(k)}$, $b = b^{(k-1)}$,
6. $b^{(k)} \leftarrow b^{(k-1)} + (u^{(k)} - B \beta^{(k)})$.
7. End while.

The $\beta$-subproblem (10) can be solved by least squares method:
\[
\beta = K^{-1} r,
\]
where $K \in \mathbb{R}^{2m \times 1}$, $K = (\Psi^T \Psi + \lambda_1 A^T A + \frac{\rho}{2} B^T B) \in \mathbb{R}^{2m \times 2m}$, $r = \frac{\rho}{2} (u + b) + \Psi^T L \in \mathbb{R}^{2m \times 1}$.

The $u$-subproblem (11) has a closed form solution for each $u_i$ (see [55])
\[
u_i = \text{shrink}((B \beta)_i - b_i, \frac{\lambda_2}{\rho}),
\]
where $\text{shrink}(a, b) = \text{sign}(a) \max(|a - b|, 0)$ and $0 \cdot (0/0) = 0$ is assumed.

### 3.2 The iterative AHF method based on the proposed model

Even though model (6) takes different behaviors of smooth and non-smooth components into consideration, we still see some blur in the high-resolution image. We find that the difference image $L - DH^{(1)}$, where $H^{(1)}$ is the resulting high-resolution output by model (6) and $D$ is a downsampling operator, contains some residual edges. To pick up more edges and details and thus to make the results less blurry, we consider the difference $L - DH^{(1)}$ as a new low-resolution input $L$ of model (6) to recompute a residual high-resolution image $H^{(2)}$. This process is repeated until the residual
edges are small enough. The sum of $H^{(1)}$ and its residual high-resolution images is the resulting super-resolution image. This iterative strategy can recover more image details (see Fig. 4). The following Algorithm 2 is the proposed iterative AHF algorithm for single image super-resolution. We use bicubic downsampling $D$ in our experiments although Algorithm 2 can work for general downsampling.

Algorithm 2 (Single image super-resolution via iterative AHF method)

**Input:**
- one low-resolution image (vector-form): $L \in \mathbb{R}^{n \times 1}$, $\lambda_1 > 0$, $\lambda_2 > 0$, $s$: upscaling factor.
- $\tau$: maximum number of iterations.

**Output:**
- high-resolution image (vector-form): $\hat{H} \in \mathbb{R}^{N \times 1}$

1. Construct matrices $\Psi_1, \Psi_2 \in \mathbb{R}^{n \times m}$ on coarse grids and $\tilde{\Psi}_1, \tilde{\Psi}_2 \in \mathbb{R}^{N \times m}$ on fine grids (see Section 3.1), where $N = s^2 n$.
2. Initialization: $L^{(1)} = L$.
   for $k = 1: \tau$
   a. Compute the coefficients: $(\beta_1^{(k)}, \beta_2^{(k)}) = \arg\min \|L^{(k)} - \Psi_1 \beta_1 - \Psi_2 \beta_2\|^2 + \lambda_1 \|\beta_1\|^2 + \lambda_2 \|\beta_2\|^1$.
   b. Update the high-resolution image: $H^{(k)} = S^{(k)} + E^{(k)}$ where $S^{(k)} = \tilde{\Psi}_1 \beta_1^{(k)}$, $E^{(k)} = \tilde{\Psi}_2 \beta_2^{(k)}$.
   c. Downsampling $H^{(k)}$ to the coarse grid: $\tilde{L} = DH^{(k)}$.
   d. Compute residual: $L^{(k+1)} = L^{(k)} - \tilde{L}$.
   end
3. Assemble the high-resolution outputs: $S = \sum_{i=1}^\tau S^{(i)}$, $E = \sum_{i=1}^\tau E^{(i)}$.
4. Compute the final high-resolution image:
   $\hat{H} = S + c(E, p)$, where $c$ represents a convolution operator, and $p$ is a Gaussian kernel with small size.

Note that step 2.a in Algorithm 2 is solved by Algorithm 1. As shown in Fig. 5, $E$ represents some edges. In particular, we use a Gaussian kernel $p$ with size $5 \times 5$ and standard derivation 1 to make a convolution to reduce the oversharp information of the non-smooth component $E$ on non-edge regions. In addition, although we introduce some parameters in the algorithm, they are not sensitive and easy to select. We will give a remark on parameter selection in Section 4.

3.3 Apply the proposed method to image patches

We have proposed the iterative AHF method for single image super-resolution, but it can be very expensive if we apply it to a whole image. For instance, if one low-resolution image has size $100 \times 100$ and the upscaling factor is 2, then $\Psi_1, \Psi_2$ with 10 angles should have size $10,000 \times 100,000$. It is very expensive to implement Algorithm 2 with large-scale non-sparse matrices. In this work, we apply
Figure 5: Illustrating step 4 of Algorithm 2. (a) The sum image $S + c(E, p)$; (b) the image $S$; (c) the image $c(E, p)$ which contains some edges and high-frequency information away from edges; (d) the image $E$ without using the convolution operator. Note that the image $E$ contains more oversharp information on non-edge regions than the image $c(E, p)$. For better visualization, we add 0.3 to the intensities of $c(E, p)$ and $E$ to brighten them.

the iterative AHF method to image patches, so that we can reduce computation and storage size significantly. We set patch size to be $6 \times 6$ and overlap to be 2 pixels in our work. We take the mean as the intensity of pixels in the overlapping region. Moreover, the intensity of pixels on the boundary will be handled by bicubic interpolation.

In summary, we use a set of Heaviside functions with sharp edges to represent sharp image edges which can be viewed as a kind of image high-frequency information. In particular, applying the proposed method to each image patches may make image high-frequency information more significant. This is the reason why the proposed method can pick up high-frequency information.

4 Numerical experiments

In this section, two kinds of test images are utilized to illustrate the performance of different methods. One is low-resolution images without high-resolution ground-truth. The other is low-resolution images downsampled from known high-resolution images, for which we can make quantitative comparisons. All experiments are implemented in MATLAB(R2010a) on a laptop of 3.25Gb RAM and Intel(R) Core(TM) i3-2370M CPU: @2.40 GHz.

We mainly compare the proposed method with some competitive image super-resolution methods: nearest-neighbor interpolation, bicubic interpolation, a kernel regression method (denoted as “07’TIP” [49]), a fast upsampling method (denoted as “08’TOG” [44]), TV-based super-resolution method (denoted as “11’JMIV” [8]), two state-of-the-art learning-based methods (“10’TIP” [59] and “12’BMVC” [3]) and three state-of-the-art interpolation-based methods (denoted as “11’IPOL” [25], “11’SIAM” [24] and “14’TIP” [44]).

For gray images, we can apply the proposed Algorithm 2 directly. For color images such as RGB, we first transform from RGB color space to “Ycbcr” color space which is very popular in image/video processing. “Y” represents luma component, “Cb” and “Cr” are blue-difference and red-difference component, respectively. We only apply the proposed algorithm to the illuminance channel because humans are more sensitive to luminance changes. In addition, we interpolate the color layers (Cb, Cr) using bicubic interpolation. After getting the upsampled images in “Ycbcr” color space, we transform them back to the original RGB color space for visualization.

In the experiments, we use three measures: root-mean-square error (RMSE), peak signal-noise ratio (PSNR) and the structural similarity (SSIM) index\footnote{https://ece.uwaterloo.ca/~z70wang/research/ssim/}, to evaluate quantitative performance.

**Remark on parameter selection:** For the parameters in Algorithm 1 and Algorithm 2, we set $\rho = 10^{-4}$, $\lambda_1 = 10^{-2}$, $\lambda_2 = 10^{-6}$, $\tau = 3$. Matrices $\Psi_1, \bar{\Psi}_1$ with $\xi = 10^{-1}$ and $\Psi_2, \bar{\Psi}_2$ with $\xi = 10^{-4}$. We set 12 angles evenly distributed on $[0, 2\pi]$, i.e., $\theta = \{0, \frac{\pi}{6}, \frac{\pi}{3}, \frac{2\pi}{3}, \frac{5\pi}{6}, \pi, \frac{7\pi}{6}, \frac{4\pi}{3}, \frac{3\pi}{2}, \frac{5\pi}{3}, \frac{11\pi}{6}\}$. We
fix $q$ as the total number of pixels of the low-resolution patch, i.e., $q = 36$ for $6 \times 6$ patches. Note that, fine tuning of parameters for some images may get better results. However, we unify the parameter selection to illustrate the stability of the proposed method.

4.1 Super-resolution for generic images

First, we test on low-resolution images without known high-resolution images (see Fig. 6). We compare the proposed method with bicubic interpolation, two state-of-the-art learning-based methods (“10’TIP” 59 and “12’BMVC” 3), a fast upsampling method (“08’TOG” 44) and three state-of-the-art interpolation-based methods (“11’IPOL” 23, “11’SIAM” 24 and “14’TIP” 54). The upsampling factors are set to be 3. From the resulting images, super-resolution images by bicubic interpolation generate a blur effect. “08’TOG” performs well on image edges, but it smoothens the intensities in non-edge regions. The two learning-based methods (“10’TIP” and “12’BMVC”) lead to visually comparable results. In addition, “11’IPOL” and “11’SIAM” also show similar performance. In particular, “14’TIP” appears to be better on image edges, but it smoothens out image details on non-edge regions. The proposed method outperforms all the others, especially for image details on non-edge regions. In Fig. 7 and Fig. 8, low-resolution test images are generated by downsampling known high-resolution via bicubic interpolation. Since ground-truth images are known, we can compute quantitative measures such as RMSE, PSNR and SSIM. The upscaling factors are set to be 4 for “baboon” and “forest”, 3 for “field” and 2 for “leopard”. From these figures, one can see that the results of “08’TOG” generate sharp edges but smoothen the intensities on non-edge regions. TV-based method “11’JMIV” 8 also obtains sharp edges but has oil-painting effect (e.g., see the close-up of “baboon” in Fig. 7). Nearest-neighbor interpolation in Fig. 8 generates a significant jaggy effect. Results of bicubic interpolation and the kernel regression method “07’TIP” 49 show significant blur. Since the proposed method can be viewed as an interpolation method, we compare it with three state-of-the-art interpolation-based methods (“11’IPOL” 23, “11’SIAM” 24 and “14’TIP” 54). In addition, we also compare the proposed method with two modern learning-based methods (“10’TIP” 59 and “12’BMVC” 3). As shown in Fig. 7 and Fig. 8, the proposed method obtains the best quantitative results for almost all test examples (see captions of the two figures), and generates high-resolution images with more image details. Actually, learning-based methods need extra training data which can be viewed as extra prior information while our method does not require any extra data. Note that the interpolation-based method “14’TIP” 54 keeps very sharp image edges and runs quite fast, but it does not obtain good quantitative results and smoothens out image details significantly on non-edge regions (see close-ups for better visualization). Furthermore, more quantitative results can be found in Tab. 2 that also demonstrates the effectiveness of the proposed method.

4.2 Super-resolution for images with smooth background

The proposed method performs well for natural images. It is a better method than the compared methods. However, the proposed method encounters a drawback: ring artifacts along large scale edges of images with smooth background (see Fig. 9(b)). The images with smooth background are generally obtained from single lens reflex (SLR) cameras or other specific scenes. The ring artifacts mainly come from the added non-smooth component (see step 4 of Algorithm 2). In this section, we utilize a method without ring artifacts to make a mask, aiming to discard the ring artifacts of non-smooth component $E$. We learn that bicubic interpolation will not cause ring artifacts (see the resulting images in section 4.1) and runs very fast. Thus we use bicubic interpolation as our intermediate method to make the mask. Actually, we only need to update the $E$ in the step 4 of Algorithm 2 by $E_{\text{new}}$ which is obtained from the following equations, then we can reduce the ring artifacts significantly.

$$E_{\text{new}} = M * E,$$

where $M$ is the mask matrix defined as follows,

$$M = \begin{cases} 
0, & \text{if } 0 \leq G_{i,j} \leq t, \\
1, & \text{otherwise}.
\end{cases}$$
Figure 6: Results of “butterflywing” and “landscape”, upscaling factors are all 3. From left to right and top to bottom: low-resolution images, results of bicubic interpolation, “10’TIP” [59], “08’TOG” [44], “11’IPOL” [25], “11’SIAM” [24], “12’BMVC” [8], “14’TIP” [54] and the proposed method (color images are better visualized in the pdf file).
Figure 7: Visual results and the corresponding (RMSE, PSNR, SSIM) of “baboon” and “forest” by the upscaling factor of 4. *First example*: bicubic interpolation (19.69, 22.25, 0.704), “08’TOG” (19.54, 22.31, 0.714), “11’JMIV” (19.81, 22.19, 0.720), “11’IPOL” (19.32, 22.41, 0.760), “11’SIAM” (19.28, 22.43, 0.765), “12’BMVC” (19.28, 22.43, 0.749), “14’TIP” (21.51, 21.48, 0.591) and the proposed method (19.20, 22.45, 0.760). *Second example*: bicubic interpolation (18.41, 22.83, 0.687), “08’TOG” (18.10, 22.98, 0.701), “11’JMIV” (18.56, 22.76, 0.700), “11’IPOL” (18.23, 22.92, 0.739), “11’SIAM” (18.02, 23.01, 0.750), “12’BMVC” (17.99, 23.03, 0.735), “14’TIP” (20.55, 21.88, 0.553) and the proposed method (17.81, 23.09, 0.758) (Bold number: the best one; Underlined number: the second best).
Figure 8: Visual results of “field” (upsampling factor 3) and “leopard” (upsampling factor 2), and the corresponding (RMSE, PSNR, SSIM). First example: nearest-neighbor interpolation (13.01, 25.52, 0.579), bicubic interpolation (12.56, 26.20, 0.599), “07’TIP” (14.70, 25.12, 0.551), “08’TOG” (12.37, 26.32, 0.595), “10’TIP” (12.33, 26.36, 0.615), “11’IPOL” (12.28, 26.39, 0.638), “11’SIAM” (12.17, 26.50, 0.646), “12’BMVC” (12.17, 26.48, 0.631), “14’TIP” (13.86, 25.37, 0.559) and the proposed method (12.08, 26.53, 0.642). Second example: “07’TIP” (20.25, 22.51, 0.772), “08’TOG” (16.44, 23.81, 0.832), “11’IPOL” (14.65, 24.81, 0.850), “11’SIAM” (14.40, 25.00, 0.855), “12’BMVC” (14.28, 25.01, 0.849), “14’TIP” (18.46, 22.84, 0.779) and the proposed method (14.23, 25.11, 0.858).
Table 2: RMSE, PSNR and SSIM for more test examples (Bold: the best one; Underlined: the second best).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>lena(X2)</td>
<td>RMSE</td>
<td>5.10</td>
<td>9.91</td>
<td>6.98</td>
<td>4.09</td>
<td>3.84</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>33.52</td>
<td>28.11</td>
<td>31.07</td>
<td>36.02</td>
<td>37.51</td>
</tr>
<tr>
<td></td>
<td>SSIM</td>
<td>0.961</td>
<td>0.920</td>
<td>0.941</td>
<td>0.982</td>
<td>0.995</td>
</tr>
<tr>
<td>landscape(X2)</td>
<td>RMSE</td>
<td>10.75</td>
<td>13.55</td>
<td>11.74</td>
<td>10.09</td>
<td>9.63</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>27.57</td>
<td>25.47</td>
<td>27.25</td>
<td>27.84</td>
<td>28.55</td>
</tr>
<tr>
<td></td>
<td>SSIM</td>
<td>0.905</td>
<td>0.690</td>
<td>0.893</td>
<td>0.916</td>
<td>0.929</td>
</tr>
<tr>
<td>purplebutterfly(X2)</td>
<td>RMSE</td>
<td>13.03</td>
<td>16.85</td>
<td>14.19</td>
<td>11.69</td>
<td>11.43</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>25.14</td>
<td>23.71</td>
<td>25.30</td>
<td>27.26</td>
<td>27.35</td>
</tr>
<tr>
<td></td>
<td>SSIM</td>
<td>0.802</td>
<td>0.759</td>
<td>0.781</td>
<td>0.865</td>
<td>0.871</td>
</tr>
<tr>
<td>leaf(X3)</td>
<td>RMSE</td>
<td>17.51</td>
<td>18.23</td>
<td>17.55</td>
<td>17.39</td>
<td>17.28</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>22.12</td>
<td>21.11</td>
<td>22.04</td>
<td>22.85</td>
<td>23.01</td>
</tr>
<tr>
<td></td>
<td>SSIM</td>
<td>0.726</td>
<td>0.715</td>
<td>0.720</td>
<td>0.733</td>
<td>0.741</td>
</tr>
<tr>
<td>dog(X3)</td>
<td>RMSE</td>
<td>2.12</td>
<td>3.76</td>
<td>1.95</td>
<td>1.93</td>
<td>1.85</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>41.88</td>
<td>38.13</td>
<td>42.51</td>
<td>42.57</td>
<td>43.00</td>
</tr>
<tr>
<td></td>
<td>SSIM</td>
<td>0.997</td>
<td>0.996</td>
<td>0.998</td>
<td>0.998</td>
<td>0.999</td>
</tr>
<tr>
<td>fruits(X3)</td>
<td>RMSE</td>
<td>4.13</td>
<td>9.09</td>
<td>3.55</td>
<td>3.75</td>
<td>3.53</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>36.11</td>
<td>29.55</td>
<td>38.61</td>
<td>38.00</td>
<td>38.66</td>
</tr>
<tr>
<td></td>
<td>SSIM</td>
<td>0.992</td>
<td>0.948</td>
<td>0.996</td>
<td>0.995</td>
<td>0.996</td>
</tr>
<tr>
<td>babyface(X3)</td>
<td>RMSE</td>
<td>5.12</td>
<td>8.44</td>
<td>4.86</td>
<td>4.87</td>
<td>4.74</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>33.60</td>
<td>29.43</td>
<td>33.91</td>
<td>33.91</td>
<td>33.98</td>
</tr>
<tr>
<td></td>
<td>SSIM</td>
<td>0.957</td>
<td>0.925</td>
<td>0.962</td>
<td>0.962</td>
<td>0.964</td>
</tr>
<tr>
<td>train(X4)</td>
<td>RMSE</td>
<td>12.98</td>
<td>15.85</td>
<td>12.64</td>
<td>12.85</td>
<td>12.62</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>25.29</td>
<td>23.28</td>
<td>26.07</td>
<td>25.55</td>
<td>26.14</td>
</tr>
<tr>
<td></td>
<td>SSIM</td>
<td>0.811</td>
<td>0.763</td>
<td>0.829</td>
<td>0.818</td>
<td>0.831</td>
</tr>
</tbody>
</table>

where $G_{i,j}$ is the vector norm of gradient at the location $(i,j)$ of image $B$; and the image $B$ is a high-resolution image generated by bicubic interpolation with an upscaling factor $s$; notation “∗” here stands for dot product; $t$ is a thresholding value, and we set $t = 0.05$ in our work. Obviously, this strategy reduces the ring artifacts of super-resolution images (see Fig. 9(c)).

In Fig. [10] we apply the proposed method to images “comic” and “face” which have smooth background. The upscaling factors are set to be 2 and 4, respectively. We can find more details from the close-ups in Fig. [10]. The proposed method reduces ring artifacts significantly and performs competitively, both visually and quantitatively.

In Fig. [11] we compare the proposed method with some competitive methods. The upscaling factor is 2. The test image “butterfly” has obvious smooth background. The results of bicubic interpolation show a significant blur effect. Although the results by “08’TOG” and “14’TIP” keep sharp edges well, they oversmoothen image details of non-edge regions. The two learning-based methods “10’TIP” and “12’BMVC” obtain competitive visual and quantitative results (see the figure caption for quantitative results). In addition, the two interpolation methods “11’IPOL” and “11’SIAM” generate clear visual results and run very fast. In particular, the proposed method preserves image details of non-edge regions well and overcomes ring artifacts significantly. Furthermore, the method also yields competitive quantitative results. Please note that the quantitative results by the proposed method in Fig. [10] and Fig. [11] are not the best. Although the updating strategy in Eq. (14) can reduce ring artifacts significantly, it however discards some crucial information in $E$, e.g., image details away from edges. This will lead to not so good quantitative results.
Figure 9: Results of “butterfly” by an upscaling factor of 2. Image “butterfly” is an image with smooth background: (a) low-resolution image; (b) super-resolution image by the proposed method using original $E$; (c) super-resolution image by the proposed method using updated $E_{\text{new}}$.

Figure 10: Results of “comic” (first row, upscaling factor 2) and “face” (second row, upscaling factor 4), and the corresponding (RMSE, PSNR, SSIM). First example: “10’TIP” (11.11, 27.15, 0.886), “11’SIAM” (10.85, 27.41, 0.896), “12’BMVC” (10.83, 27.41, 0.891) and the proposed method (11.01, 27.25, 0.888). Second example: “10’TIP” (6.59, 31.76, 0.759), “11’SIAM” (6.22, 32.26, 0.780), “12’BMVC” (6.39, 32.15, 0.772) and the proposed method (6.36, 32.17, 0.773).
Figure 11: Results of “butterfly” by an upscaling factor of 2. First row: ground-truth, results of bicubic interpolation (6.36, 32.05, 0.956), “08’TOG” (8.08, 29.97, 0.942). Second row: results of “10’TIP” (5.14, 33.86, 0.964), “11’IPOL” (5.32, 33.61, 0.966) and “11’SIAM” (5.22, 33.78, 0.967). Third row: results of “12’BMVC” (4.90, 34.33, 0.969), “14’TIP” (9.47, 28.58, 0.928) and the proposed method (5.15, 33.81, 0.967).
4.3 Discussions

**More comparisons:** We further compare the proposed method with three recent state-of-the-art methods, i.e., “09’ICCV” [11], “13’ICCV” [51] and “14’ECCV” [15]. The proposed method can actually be viewed as interpolation-based method. It does not need any extra information for training, and is a completely single image super-resolution method. The methods “13’ICCV” and “14’ECCV” are two competitive learning based methods. In particular, “14’ECCV” utilizes the recent most efficient technique: deep learning. They first utilize extra training data to learn dictionaries, and then apply the dictionaries to get the final super-resolution images. Furthermore, the method “09’ICCV” is a very novel and efficient single image super-resolution method. From Fig. 12, we can observe that the proposed method performs comparably with the others, including image edges and the details on non-edge regions.

**More on parameters selection:** The proposed method involves many parameters, e.g., $\lambda_1$, $\lambda_2$, $\xi_1$, $\xi_2$, patch size, etc. However, they are easy to select because the results are not sensitive to the selection of parameters (see $\xi_1$, $\xi_2$ in Tab. 3 and $\lambda_1$, $\lambda_2$ in Tab. 4). Choosing suitable parameters is a common difficulty for many algorithms. Tuning empirically is a popular way for determining parameters. In our work, we obtain the parameters empirically. For instance, for $\lambda_1$ and $\lambda_2$, we first fix the parameter $\lambda_1$, and tune $\lambda_2$ with 10 times change. In this work, we fix $\lambda_1 = 10^{-2}$, then tune $\lambda_2$ by $10^{-3}$, $10^{-4}$, $10^{-5}$, $10^{-6}$, $10^{-7}$, etc. When we find $\lambda_2 = 10^{-6}$ is the best one for $\lambda_1$, then tune $\lambda_2$ in the range of $[10^{-5}, 10^{-7}]$, e.g., $5 \times 10^{-6}$ and $5 \times 10^{-7}$. Actually, the results around $\lambda_1 = 10^{-2}$ and $\lambda_2 = 10^{-6}$ also do not show obvious difference (see Tab. 4). Fine tuning of parameters might lead to slightly better results (see Tab. 5). But to make it simple, we set $\lambda_1 = 10^{-2}$, $\lambda_2 = 10^{-6}$ in all test examples. Under this setting, the re-
results are already good enough for comparisons. We choose values for other parameters in a similar way.

Table 3: RMSE around $\xi_1 = 10^{-1}$ and $\xi_2 = 10^{-4}$. Test example is “field” in Fig. 8.

<table>
<thead>
<tr>
<th>$\xi_2$</th>
<th>0.8 $\times$ 10$^{-1}$</th>
<th>1 $\times$ 10$^{-1}$</th>
<th>1.2 $\times$ 10$^{-1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8 $\times$ 10$^{-4}$</td>
<td>12.09</td>
<td>12.08</td>
<td>12.07</td>
</tr>
<tr>
<td>1 $\times$ 10$^{-4}$</td>
<td>12.10</td>
<td>12.08</td>
<td>12.07</td>
</tr>
<tr>
<td>1.2 $\times$ 10$^{-4}$</td>
<td>12.09</td>
<td>12.08</td>
<td>12.08</td>
</tr>
</tbody>
</table>

Table 4: RMSE around $\lambda_1 = 10^{-2}$ and $\lambda_2 = 10^{-6}$.

<table>
<thead>
<tr>
<th>$\lambda_2$</th>
<th>0.8 $\times$ 10$^{-2}$</th>
<th>1 $\times$ 10$^{-2}$</th>
<th>1.2 $\times$ 10$^{-2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8 $\times$ 10$^{-6}$</td>
<td>12.09</td>
<td>12.08</td>
<td>12.08</td>
</tr>
<tr>
<td>1 $\times$ 10$^{-6}$</td>
<td>12.09</td>
<td>12.08</td>
<td>12.08</td>
</tr>
<tr>
<td>5 $\times$ 10$^{-6}$</td>
<td>12.10</td>
<td>12.09</td>
<td>12.08</td>
</tr>
</tbody>
</table>

Table 5: RMSE of the proposed method when setting $\lambda_1$ from 5 $\times$ 10$^{-1}$ to 1 $\times$ 10$^{-3}$ and $\lambda_2$ from 1 $\times$ 10$^{-3}$ to 1 $\times$ 10$^{-7}$ (Bold: the best one; Underlined: the second best).

<table>
<thead>
<tr>
<th>$\lambda_1$</th>
<th>5 $\times$ 10$^{-1}$</th>
<th>1 $\times$ 10$^{-1}$</th>
<th>5 $\times$ 10$^{-2}$</th>
<th>1 $\times$ 10$^{-2}$</th>
<th>5 $\times$ 10$^{-3}$</th>
<th>1 $\times$ 10$^{-3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 $\times$ 10$^{-3}$</td>
<td>12.16</td>
<td>12.14</td>
<td>12.13</td>
<td>12.11</td>
<td>12.13</td>
<td>12.18</td>
</tr>
<tr>
<td>1 $\times$ 10$^{-4}$</td>
<td>12.13</td>
<td>12.12</td>
<td>12.11</td>
<td>12.10</td>
<td>12.12</td>
<td>12.17</td>
</tr>
<tr>
<td>1 $\times$ 10$^{-5}$</td>
<td>12.11</td>
<td>12.10</td>
<td>12.10</td>
<td>12.09</td>
<td>12.11</td>
<td>12.15</td>
</tr>
<tr>
<td>5 $\times$ 10$^{-6}$</td>
<td>12.09</td>
<td>12.07</td>
<td>12.08</td>
<td>12.08</td>
<td>12.11</td>
<td>12.14</td>
</tr>
<tr>
<td>1 $\times$ 10$^{-6}$</td>
<td>12.09</td>
<td>12.07</td>
<td>12.07</td>
<td>12.08</td>
<td>12.10</td>
<td>12.14</td>
</tr>
<tr>
<td>5 $\times$ 10$^{-7}$</td>
<td>12.10</td>
<td>12.08</td>
<td>12.08</td>
<td>12.10</td>
<td>12.10</td>
<td>12.14</td>
</tr>
<tr>
<td>1 $\times$ 10$^{-7}$</td>
<td>12.11</td>
<td>12.10</td>
<td>12.09</td>
<td>12.11</td>
<td>12.11</td>
<td>12.17</td>
</tr>
</tbody>
</table>

**Computation time:** From Fig. [13](a), we observe that the computation time does not increase significantly when the upscaling factor is increased from 2 to 9. Using a non-optimized Matlab code, it is about below 10 seconds for a low-resolution image 60 $\times$ 60. From Fig. [13](b), when the size of low-resolution image is from 30 $\times$ 30 to 110 $\times$ 110, the computation time is increased from 3 seconds to 34 seconds. We can conclude that the computation time mainly depends on the size of low-resolution image but not so significantly on the upscaling factor, since the main computation is to compute coefficients $\beta_1$ and $\beta_2$ via the low-resolution image. Note that, there is a lot of room to speed up the speed. We can use cmex in Matlab to speed up the code that involves a lot of loops. We can also use parallel computing as the computation is done patch by patch.

**About $\theta$ and $c$:** The proposed method is based on the approximated Heaviside functions that mainly depend on two factors $\theta$ and $c$ (see Eq. (4)). Here, we tend to study the relations on RMSE, computation time and parameter pairs. From the first row of Fig. [14] the RMSE of the proposed method decreases smoothly (only about 0.05 RMSE reduction) with $q$ increasing from 36 to 90 when setting $k = 12$, while computation time increases significantly from about 20 seconds to 60 seconds. Thus
The relation between model (6), and model (6) combined with iterative idea: It is necessary to illustrate the relation between model (6), and model(6) combined with our iterative idea. There is only fine visual difference, especially in image details and edges, between model (6), and model (6) combined with our iterative idea. There is no significant visual difference between the two methods. However, from Fig. 15, it is easy to know that the proposed model (6) combined with the iterative AHF method performs smaller RMSE than the proposed model (6). In particular, the model (6) with the iterative strategy will result in more computation.

5 Conclusions

In this paper, we casted image super-resolution problem as an intensity function estimation problem. We assumed the underlying intensity function, defined on a continuous domain and belonging to a space with redundant basis, could be approximately represented by two classes of approximated Heaviside functions (AHFs). The representation coefficients were computed by the proposed iterative AHF method using only one low-resolution image. We then applied the coefficients to get high-resolution images. To reduce computation and storage size, we applied the proposed iterative AHF method to image patches. For images with smooth background, we designed a strategy utilizing image gradient to reduce ring artifacts along large scale edges. The method can be applied to any upscaling factors and needs no extra data for learning. In particular, we also discussed the parameter selection and computation time. Many experiments show that the proposed approach outperforms existing competitive methods, both visually and quantitatively.
Figure 14: The investigation on RMSE, computation time and the number of parameter pairs \((k, q)\). First row: \(k = 12\), the RMSE of the proposed method decreases smoothly with the number of pixels \(q\) increasing from 36 to 90 (left), whereas the computation time increases significantly (right). Second row: \(q = 36\), the RMSE of the proposed method decreases smoothly with the number of directions \(k\) increasing from 12 to 60 (left), while the computation time increases significantly (right). It demonstrates that there is no significant RMSE reduction with more parameter pairs \((k, q)\), but increasing computation time apparently.
Figure 15: First column: results of model (6) with iterative AHF method. Second column: results of model (6) without iterative AHF method. Third column: error maps (for better vision, we add an extra intensity of 30/255 to the true error maps). Upscaling factors: 4.
Acknowledgement

The first and third authors thank the support by 973 Program (2013CB329404), NSFC (61370147), Sichuan Province Sci. & Tech. Research Project (2012GZX0080). The first author is also supported by NSFC (61402082), Fundamental Research Funds for the Central Universities and Outstanding Doctoral Students Academic Support Program of UESTC. The second author thanks US NIH 1R21EB016535-01 and NSF DMS-1521582 for partial support.

References


