PLOS ONE

RESEARCH ARTICLE

Single image super-resolution based on approximated Heaviside functions and iterative refinement

Xin-Yu Wang, Ting-Zhu Huang*, Liang-Jian Deng*

School of Mathematical Sciences/Research Center for Image and Vision Computing, University of Electronic Science and Technology of China, Chengdu, Sichuan, P. R. China

* tingzhuhuang@126.com (TZH); liangjian1987112@126.com (LJD)

Abstract

One method of solving the single-image super-resolution problem is to use Heaviside functions. This has been done previously by making a binary classification of image components as "smooth" and "non-smooth", describing these with approximated Heaviside functions (AHFs), and iteration including l_1 regularization. We now introduce a new method in which the binary classification of image components is extended to different degrees of smoothness and non-smoothness, these components being represented by various classes of AHFs. Taking into account the sparsity of the non-smooth components, their coefficients are l_1 regularized. In addition, to pick up more image details, the new method uses an iterative refinement for the residuals between the original low-resolution input and the down-sampled resulting image. Experimental results showed that the new method is superior to the original AHF method and to four other published methods.

Citation: Wang X-Y, Huang T-Z, Deng L-J (2018) Single image super-resolution based on approximated Heaviside functions and iterative refinement. PLoS ONE 13(1): e0182240. https://doi.org/10.1371/journal.pone.0182240

Editor: Li Zeng, Chongqing University, CHINA

Received: May 30, 2016

Accepted: December 28, 2017

Published: January 12, 2018

Data Availability Statement: All relevant data are within the paper.

Funding: The authors received no specific funding for this work.

Competing interests: The authors have declared that no competing interests exist.

Introduction

Image super-resolution (SR) is to generate or recover high-resolution (HR) images from *one* or *multiple* low-resolution (LR) images. If we generate/recover the HR image from only one LR image, we call it *single-frame* SR. Otherwise, we call it *multiple-frame* SR. The multiple-frame SR methods are available in that multiple LR images are of the same scene with different sub-pixel shifts taken as input. It has direct applications in video SR problems (see [1]). *Single-frame* SR methods are quite popular and challenging when only one LR image is avaliable. In particular, we focus on *single-frame* SR problems in this paper.

To produce a HR image, the simplest and effective way is to interpolate, e.g., bicubic and nearest interpolations. Recently, more interpolation-based methods have been proposed(see [2–8]). For instance, Youngjoon et al. [8] utilize a generalized curvature source term estimated from the LR image to construct a HR image. In particular, the resulting HR image has a reliable curvature profile which minimizes ringing artifacts. In addition, they propose an iterative application of the curvature interpolation method [9]. The method utilizes the gradient-weighted curvature measured from the LR image, being an interpolator to suppress texture oversmoothing. In [10], Wang et al. present a fast image upsampling method to preserve the

sharpness via two-scale sharpness preserving operations. On the one hand, the low-frequency of image is recovered based on a well-constructed displacement field. On the other hand, the local high-frequency structures are reconstructed via a sharpness preserving reconstruction algorithm. However, due to images zooming with the interpolation-based methods to be solved are frequently estimated by information of unknown locations without other priors. The interpolation-based methods usually introduce jagged artifacts or blur effect.

Learning-based methods (see [11–14]) have attracted attention in image processing. Recently, they are also performed impressively in image SR problems. A key issue for the learning-based methods is to learn high frequency correspondences from a database generated by LR and HR image patches pairs. And then we could apply the correspondences to the LR image input to obtain its HR output. Purkait et al. [11] develop fuzzy rules to find different possible HR patches and combine them according to different rule strength to obtain the estimated HR patches. The rule parameters are learned from LR-HR patch pairs and then they use the Takagi-Sugeno (TS) model [15] with the rule parameters expressed as a linear combination of the different input possible HR patches. In addition, Yang et al. [12] apply the theory of sparse coding to SR problems effectively. This method jointly trains two dictionaries for LR and HR image patches, and then they could use the LR dictionary to generate sparse representations of the LR input to obtain the corresponding HR output. And Dong and Loy [13] first learn a mapping between low-resolution and high-resolution images, which is represented by a deep convolutional neural network (CNN). And then they take the LR image as input via the CNN to generate the HR image output. However, these methods typically reply on the similarity between test images and the database. Consequencely, they also involve expensive computation. In recent years, there has been tremendous interest in developing statistics-based methods [14, 16, 17], such as the popular tool Maximum a Posteriori (MAP) and Maximum Likelihood estimator (MLE). As proposed in [14], Peleg and Elad assume that prediction of high resolution patches can be obtained by MMSE estimation and the resulting scheme has the useful interpretation of a feedforward neural network.

Apart from the methods mentioned above, a hybrid method [18], reconstruction methods [19–21], and other methods [22–25] have been used. And these methods are not completely independent with each other. For instance, Peleg and Elad propose a SR method via combining a statistics method and a learning-based method.

In this paper, we study an effective single-frame SR approach, which is an improvement of the so called approximated Heaviside functions method (AHFM) proposed in [7]. It shows that the underlying image, can be viewed as a instensity function, which can be approximately represented by two classes of AHFs. Deng et al. cast the image super-resolution problem as an intensity function estimation problem. Defined on a continuous domain, the underlying intensity image, which belongs to a space with redundant basis, could be approximately represented via two classes of AHFs. Using only one LR image input, we could compute the representation coefficients by the proposed iterative AHF method. Then the high-resolution image is generated by combining the representation coefficients with two classes of AHFs. Here, the two classes of AHFs are corresponding to the high-resolution image.

However, there are only two classes of AHFs, which may not describe/represent the whole information of one image. It may generate some oversharp information for the final HR image. Forced by that, we tend to extend the AHFM algorithm to general form to suppress the resulted image texture oversharp and improve super-resolution images qualities. We consider that an image is normally consisted of different smooth components and non-smooth components. In particular, there are some details, such as edges and corners, which have different sharpness. Based on this, we propose that the smooth components should be represented by multiple classes of AHFs with smooth edges, and the non-smooth components are also

represented by multiple classes of AHFs with sharp edges. In particular, due to the sparsity of non-smooth components, we give l_1 regularization model and solve the model via block-wise alternating direction method of multipliers (ADMM) [26]. Furthermore, we design a novel iterative refinement algorithm to pick up more image details. Finally, the proposed method has been numerically proved competitively to some state-of-the-art methods.

The paper is organized as follows. A brief review of the AHFM algorithm [7] is introduced in Section 2. In Section 3, we give the proposed model and its corresponding algorithms. In Section 4, we present the numerical results of different methods. It demonstrates that the proposed algorithms are more efficient. Finally, we conclude the paper in Section 5.

Some preliminaries

This section gives general remarks on approximated Heaviside functions. In addition, a brief review of the method based on approximated Heaviside functions can be found from [7].

2.1 General remarks on Heaviside function

Heaviside function or Heaviside step function, is a discontinuous function whose value is zero for negative argument and one for positive argument. Heaviside function could be defined as following alternative form of the unit step, as a function of a discrete variable x (see Fig 1(a)),

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

The definition of $\phi(0) = 0$ is significant. In the practical applications, some logistic functions to the Heaviside functions are often used for smooth approximations, called approximated Heaviside functions (AHFs), such as

$$\psi(x) = \frac{1}{1 + e^{-2x/\xi}}, \tag{2}$$

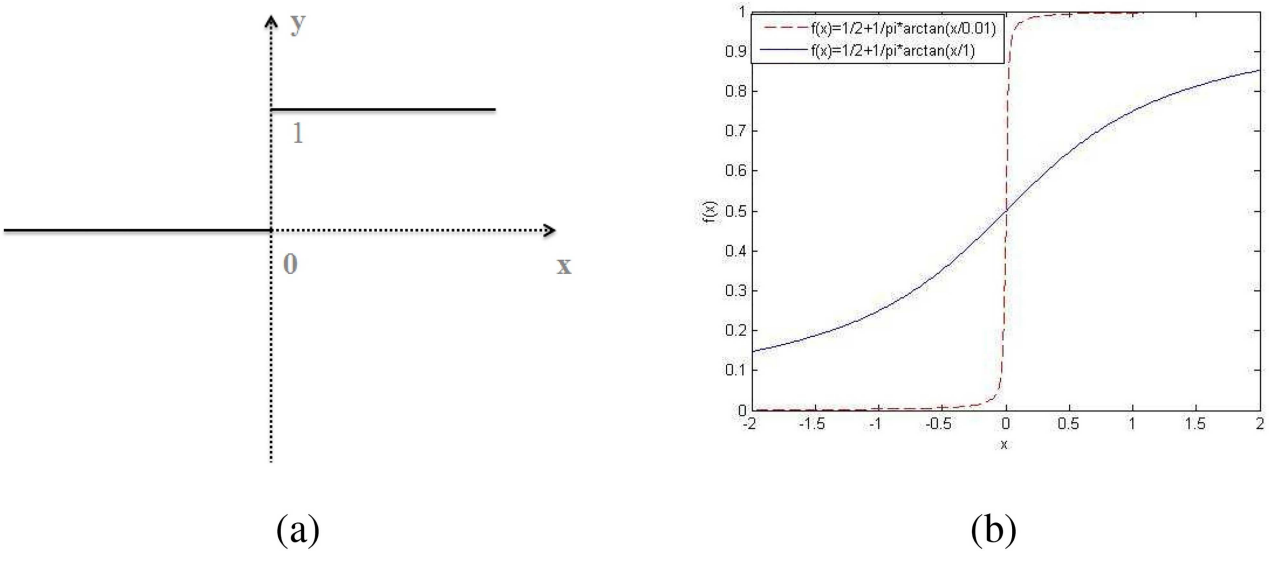

Fig 1. (a) Heaviside function. (b) two approximated Heaviside functions with $\xi = 1$ (blue solid line) and $\xi = 0.01$ (red dash line).

https://doi.org/10.1371/journal.pone.0182240.g001

or

$$\psi(x) = \frac{1}{2} + \frac{1}{\pi}\arctan\left(\frac{x}{\xi}\right). \tag{3}$$

As illustrated in Fig 1(b), a smaller ξ corresponds to a sharper transition at $x = 0$. In our work, we employ Eq (3) to approximate Heaviside functions.

From [27], Kainen et al. propose that any functions in $L_p([0, 1]^d)$, $p \in [1, \infty)$ could be well approximated by linear combinations of m characteristic functions of half-space, and 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 ψ is an approximated functions. H_d is the set of characteristic functions of closed half-space of $\mathcal{R}^d$.

Theorem 1 [27] For any positive integer d, define $span_m H_d$ as $\{\sum_{i=1}^m \omega_i \psi(v_i \cdot x + c)\}$, where $\omega_i \in \mathcal{R}$ and $v_i \in \mathcal{R}^d$, and $c_i \in \mathcal{R}$, then it is known that $U_{m \in n^+} span_m H_d$ is dense in $(L_p([0, 1]^d), \|\cdot\|_p)$, $p \in [1, \infty)$.

Theorem 2 [27] For any positive integer m,d and every $p \in [1, \infty)$, $span_m H_d$ is approximately a compact sub-set of $(L_p([0, 1]^d), \|\cdot\|_p)$.

Consequently, we can use $span_m H_d$ for a finite m in practical computing.

2.2 Single image super-resolution via iterative AHF method (AHFM)

The single image super-resolution via iterative AHF method (AHFM) proposed in [7] for image super-resolution gives a selection of sharp-related terms, which are measured from the LR image input and apply them to fine grids to generate the HR image. They assume the underlying image intensity function f is defined on $[0, 1]^2$, then $f \in L_p([0, 1]^2)$ with $p \in [1, \infty)$. According to the theorems stated in section 2.1, f can be approximated by the following equation:

$$f(\mathbf{z}) = \sum_{j=1}^{m} \omega_j \psi(\mathbf{v}_j \cdot \mathbf{z} + c_j), \tag{4}$$

where $\omega_j \in \mathcal{R}$, $v_j \in \mathcal{R}^2$, $\mathbf{v} = \{(cos\theta_t, sin\theta_t)', t = 1, 2, \ldots, p\}$ denote p different directions, and $c_j = \{\frac{1}{q}, \frac{2}{q}, \frac{3}{q} \ldots, 1\}$ is to denote discrete positions, $m = pq$, $\mathbf{z} = (x, y)'$, where q is the total number of pixels of the input image. Consequently, the function $\{\psi(\mathbf{v}_j \cdot \mathbf{z} + c_j)\}_{j=1}^m$ is called a class of AHF with a specific ξ. For an image $L \in \mathcal{R}^{n_1 n_2}$, we assume it is a discretization of intensity function f on $[0, 1]^2$, i.e., $L_{i,j} = f(x_i, y_j), x_i = \frac{i}{n_1}, y_j = \frac{j}{n_2} (i = 1, 2, \cdots, n_1, j = 1, 2, \cdots, n_2)$. Therefore, Eq (4) could be rewritten as matrix-vector form, $L \approx \Psi\omega, f \in R^n, \omega \in \mathcal{R}^m$, with $n = n_1 n_2$, $m = pq$. We compute coefficient ω and get the high resolution image with equation $\tilde{\Psi}\omega$, where s is an upscaling factor and $\tilde{\Psi} \in \mathcal{R}^{N \times m}$ with size $N = s^2 n_1 n_2$.

By this strategy, based on the observation that an image consists of smooth components and non-smooth components. We use two classes of AHFs to depict an image. Different components of an image may be described by different orientations θ_t at the locations c_j with two ξ_1, ξ_2. One is big parameter ξ_1 to represent smooth components (forming Ψ_1), another one is the smaller ξ_2 to represent non-smooth components (forming Ψ_2). Thus, the vector-form image L can be approximated by the following discrete formula:

$$L \approx \Psi_1 \beta_1 + \Psi_2 \beta_2. \tag{5}$$

Since non-smooth components are sparse, l_1 regularization could be given on the coefficient β_2. The optimal model is expressed as:

$$\|L - \Psi_1\beta_1 - \Psi_2\beta_2\|_2^2 + \lambda_1\|\beta_1\|_2^2 + \lambda_2\|\beta_2\|_1. \tag{6}$$

The Eq (6) is solved via ADMM [28–31] in article [7]. The single image super-resolution via iterative AHF method (AHFM) proposed in [7] is outlined as shown in Algorithm 1.

We close this section with the following remarks.

• In order to pick up more details, such as edges, they design an iterative strategy to conduct an iterative refinement. They consider the difference ($L - DH$) as a new low-resolution input of Eq (6) to recompute a residual high-resolution image.

• The AHFM algorithm performs well for natural images. However, the images with smooth backgrounds usually appear ring artifacts along the large scale edges, which mainly come from the added non-smooth components (see step (b) in Algorithm 1). Aiming to discard the ring artifacts of non-smooth components E, they use bicubic interpolation as the intermediate method to make a mask. Actually, in the step (b), only the E needs to update by E_{new}, which can be obtained from by the Eq (7), and the ring artifacts could be reduced significantly.

$$E_{new} = \text{Mask.} * E, \tag{7}$$

where

$$\text{Mask} = \begin{cases} 0, & \text{if } 0 \le G_{i,j} \le t, \\ 1, & \text{otherwise}, \end{cases} \tag{8}$$

where $G_{i,j}$ is a vector-form of gradient at location (i, j) of image B. The image B is generated via the bicubic interpolation. Notation $.^*$ stands for dot product. t is a threshold value and $t = 0.05$ is in the experiments.

Algorithm 1 (Single image super-resolution via iterative AHF method (AHFM) [7])

Input: one vector-form low-resolution image: $L \in \mathcal{R}^{n\times 1}, \lambda_1 > 0, \lambda_2 > 0$, s: upscaling factor. τ: maximum number of iteration.

Output: high-resolution image $\hat{H} \in \mathcal{R}^{N\times 1}$

1. According to Eq (4), construct matrices $\Psi_1, \Psi_2 \in \mathcal{R}^{n\times m}$ on coarse grids, on fine grids construct $\tilde{\Psi}_1, \tilde{\Psi}_2 \in \mathcal{R}^{N\times m}$, where $N = s^2 n$.
2. Initialization: $L^{(1)} = L$.

   for k = 1:τ

   1. Compute the coefficients:

      $$(\beta_1^{(k)}, \beta_2^{(k)}) = \text{argmin}\|L^{(k)} - \Psi_1\beta_1 - \Psi_2\beta_2\|_2^2 + \lambda_1\|\beta_1\|_2^2 + \lambda_2\|\beta_2\|_1.$$

   2. Update the high-resolution image:

      $$H^{(k)} = S^{(k)} + E^{(k)},$$
      $$\text{where } S^{(k)} = \tilde{\Psi}_1\beta_1^{(k)}, E^{(k)} = \tilde{\Psi}_2\beta_2^{(k)}.$$

   3. Downsampling $H^{(k)}$ to coarse grid: $\tilde{L} = DH^{(k)}$.
   4. 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:

$$\hat{H} = S + conv(E, p),$$

where *conv* represents a convolution operator, and p is a Gaussian kernel with a small size.

Modified AHFM and iterative refinement

This section presents a new image super-resolution algorithm, which is an extension of the AHFM algorithm. Note that only two classes of AHFs are not enough for the whole image components. Hence, taking into account the varying sharpness of the whole image, we will consider a modified AHFM with the different sharpness components and further present its algorithm for implementation in next section. Besides, the modified AHFM algorithm contains a new iterative refinement strategy, aiming to pick up more information about non-smooth components.

3.1 Modified AHFM with different sharpness components

The AHFM has its respective advantages, which is completely a single image super-resolution method without extra training data. The AHFM algorithm has only used two classes of AHFs to represent the whole image components. The sharper components or the smoother components are pivotal but are not represented well. Motivated by the point and the works proposed in [7], we propose a modified AHFM algorithm to comprise the whole information of one image as well as possible.

First, we assume that a low-resolution image L is consisted of smooth components and non-smooth components. We exploit different ξ to form different AHFs to describe smooth components and non-smooth components. Hence, the low-resolution image L could be approximated by the following discrete formula:

$$L \approx \sum_{i=1}^{l} \Psi_i \alpha_i + \sum_{j=1}^{k} \Phi_j \beta_j, \tag{9}$$

where l, k represent numbers of smooth components and non-smooth components, respectively. We could obtain $\Psi_i, \Phi_j \in \mathcal{R}^{n\times m}(i = 1, 2, \cdots, l; j = 1, 2, \cdots, k)$ according to Eq (4). $\Psi_i(i = 1, 2, \cdots, l)$ represents smooth components formed by larger ξ. $\Phi_j(j = 1, 2, \cdots, k)$ represents non-smooth components formed by smaller ξ. And $\alpha_i, \beta_j \in \mathcal{R}^{m\times 1}(i = 1, 2, \cdots, l; j = 1, 2, \cdots, k)$ are the corresponding representation coefficients with $\Psi_i(i = 1, 2, \cdots, l)$ and $\Phi_j(j = 1, 2, \cdots, k)$. After computing the representation coefficients, we apply them into following the equation to obtain the high-resolution image:

$$H \approx \sum_{i=1}^{l} \tilde{\Psi}_i \alpha_i + \sum_{j=1}^{k} \tilde{\Phi}_j \beta_j, \tag{10}$$

where $\tilde{\Psi}_i, \tilde{\Phi}_j \in \mathcal{R}^{N\times m}(i = 1, 2, \cdots, l; j = 1, 2, \cdots, k; N = s^2 n)$ are obtained on fine grids. s is the upscaling factor.

Since the non-smooth components, such as edges and corner details, are sparse in generic images. Hence, we apply l_1 regularization on $\beta_j(j = 1, 2, \cdots, k)$ to characterize this feature.

Thus, the optimization model can be written as following:

$$\min_{\alpha_i,\beta_j} \| L - \sum_{i=1}^{l} \Psi_i \alpha_i - \sum_{j=1}^{k} \Phi_j \beta_j \|_2^2 + \sum_{i=1}^{l} \lambda_i \| \alpha_i \|_2^2 + \sum_{j=1}^{k} \gamma_j \| \beta_j \|_1, \tag{11}$$

where $\lambda_i, \gamma_j \in \mathcal{R}(i = 1, 2, \cdots, l; j = 1, 2, \cdots, k)$ are regularization parameters.

To simplify Eq (11), we set $\mathcal{C} = (\Psi, \Phi)$, $\Psi = (\Psi_1, \Psi_2, \cdots, \Psi_l)$, $\Phi = (\Phi_1, \Phi_2, \cdots, \Phi_k)$, $u = (\alpha_1, \alpha_2, \cdots, \alpha_l, \beta_1, \beta_2, \cdots, \beta_k)^T$,

$$M_i = (\overbrace{\mathbf{0}, \cdots, \mathbf{0}, \underbrace{\mathbf{I}}_{ith}, \cdots, \mathbf{0}}^{l}, \underbrace{\mathbf{0}, \cdots, \mathbf{0}}_{k}), B_j = (\underbrace{\mathbf{0}, \mathbf{0}, \cdots, \mathbf{0}}_{l}, \overbrace{\mathbf{0}, \cdots, \underbrace{\mathbf{I}}_{jth}, \cdots, \mathbf{0}}^{k}).$$

Thus, Eq (11) could be rewritten as following:

$$\min_{u} \| L - Cu \|_2^2 + \sum_{i=1}^{l} \lambda_i \| M_i u \|_2^2 + \sum_{j=1}^{k} \gamma_j \| B_j u \|_1. \tag{12}$$

As l_1 term is not differentiable, we make a set of variable substitutions for $B_j u (j = 1, 2, \cdots, k)$ and then rewrite Eq (12) as:

$$\begin{gathered} \min_{u} \| L - \mathcal{C}u \|_2^2 + \sum_{i=1}^{l} \lambda_i \| M_i u \|_2^2 + \sum_{j=1}^{k} \gamma_j \| p_j \|_1, \\ s.t., p_j = B_j u (j = 1, 2, \cdots, k), \end{gathered} \tag{13}$$

where $p_j (j = 1, 2, \cdots, k)$ are the substitution variable. In particular, the Eq (13) is separable, w.r.t $(u, p_j)(j = 1, 2, \cdots, k)$. Many methods are used to solve the l_1 problem (see [28–33]). The work in [7] solve the l_1 problem via alternating direction method of multipliers(ADMM) [31]. Here, we solve Eq (13) via block-wise ADMM [26].

The augmented Lagrangian of Eq (13) is

$$\mathcal{L}(u, p_j, b_j) = \| L - \mathcal{C}u \|_2^2 + \sum_{i=1}^{l} \lambda_i \| M_i u \|_2^2 + \sum_{j=1}^{k} \gamma_j \| p_j \|_1 + \sum_{j=1}^{k} \frac{\tau_j}{2} \| p_j - B_j u + b_j \|_2^2, \tag{14}$$

where $b_j (j = 1, 2, \cdots, k)$ are Lagrangian multipliers with proper size. The problem of minimizing $\mathcal{L}(u, p_j, b_j)$ could be solved by iteratively and alternatively via the following subproblems:

$$u\text{-}subproblem \ : \ \min_{u} \| L - \mathcal{C}u \|_2^2 + \sum_{i=1}^{l} \lambda_i \| M_i u \|_2^2 + \sum_{j=1}^{k} \frac{\tau_j}{2} \| p_j - B_j u + b_j \|_2^2, \tag{15}$$

$$p_j\text{-}subproblem \ : \ \min \gamma_j \| p_j \|_1 + \frac{\tau_j}{2} \| p_j - B_j u + b_j \|_2^2 \ (j = 1, 2, \cdots, k). \tag{16}$$

Hence, the solution to the problem (11) is solved by block-wise ADMM, which is shown in Algorithm 2:

Algorithm 2

Input: Given the low-resolution image $L \in \mathcal{R}^{n \times 1}$, $\Psi_i, \Phi_j \in \mathcal{R}^{n \times m}$, $\lambda_i (i = 1, 2, \cdots, l)$, $\gamma_j > 0$, $\tau_j > 0 (j = 1, 2, \cdots, k)$

Output: coefficient u

1. **Initialization:** $u^{(1)} \leftarrow \mathbf{0}, p_j^{(1)} \leftarrow \mathbf{0}, b_j^{(1)} \leftarrow \mathbf{0} (j = 1, 2, \cdots, k)$

2. **while not converged do**
3. $t \leftarrow t + 1$
4. $u^{(t)} \leftarrow$ solve subproblem (15) for $p_j = p_j^{(t-1)}, b_j = b_j^{(t-1)}$
5. $p_j^{(t)} \leftarrow$ solve subproblem (16) for $u = u^{(t)}, b_j = b_j^{(t-1)}$
6. $b_j^{(t)} \leftarrow b_j^{(t)} + (p_j^{(t)} - B_j u^{(t)})$.
7. **end while**

The *u-subproblem* is a smooth quadratic problem, we can solve it by least squares method:

$$u = K^{-1}q, \tag{17}$$

where $u \in \mathcal{R}^{(k+l)m\times 1}$,

$$K = \mathcal{C}^T\mathcal{C} + \sum_{i=1}^{l}\lambda_i M_i^T M_i + \sum_{j=1}^{k}\frac{\tau_j}{2}B_j^T B_j \in \mathcal{R}^{(k+l)m\times(k+l)m},$$

$$q = \mathcal{C}^T L + \sum_{j=1}^{k}\frac{\tau_j}{2}(B_j^T p_j + B_j^T b_j) \in \mathcal{R}^{(k+l)m\times 1}.$$

The p_j-*subproblem*($j = 1, 2, \cdots, k$) has a closed form solution for each $(p_j)_i$ (see [33]),

$$(p_j)_i = shrink((B_j u)_i - (b_j)_i, \frac{\gamma_j}{\tau_j}), \tag{18}$$

where $shrink(a, b) = \text{sign}(a)\max(|a| - b, 0)$ and $0.(0/0) = 0$ is assumed.

By Algorithm 2, we have computed the representation coefficients $\alpha_i, \beta_j (i = 1, 2, \cdots, l; j = 1, 2, \cdots, k)$. We can get the high resolution image $\hat{H}$ via Eq (10).

3.2 Modified AHFM with iterative refinement

The Eq (9) takes different smooth components and non-smooth components into consideration. A natural remedy for extracting more details is to find the structure of the difference between LR image and last updated resulted image. We find that residual $R = L - D\hat{H}$, where $\hat{H}$ is the last updated HR image and D is the downsampled operator. Fortunately, we find the residual is mostly non-smooth components. To pick up more edge details to make image less blurry, we design a new iterative refinement model based on the special structure. In the model, we use two smaller ξ_3^*, ξ_4^* (forming Φ_3^*, Φ_4^*) to depict the residual image. Since the non-smooth components are also sparse in the residual. We apply l_1 regularization to the corresponding coefficients. The iterative refinement model could be written as following:

$$\min_{\beta_3^*,\beta_4^*} \| R - \Phi_3^*\beta_3^* - \Phi_4^*\beta_4^* \|_2^2 + \mu_1 \| \beta_3^* \|_1 + \mu_2 \| \beta_4^* \|_1. \tag{19}$$

β_3^*, β_4^* are the coefficients of the non-smooth components. μ_1, μ_2 are regularization parameters. Since the Eq (19) is the form of l_1 norm, we solve it by ADMM scheme. We make two variable substitutions for β_3^*, β_4^*, and rewrite Eq (19) as:

$$\begin{aligned}\min_{\beta_3^*,\beta_4^*} \| R - \Phi_3^*\beta_3^* - \Phi_4^*\beta_4^* \|_2^2 + \mu_1 \| u_1 \|_1 + \mu_2 \| u_2 \|_1,\\ s.t., u_1 = \beta_3^*, u_2 = \beta_4^*.\end{aligned} \tag{20}$$

We rewrite Eq (20) as:

$$\min_{\beta^*} \| R - \mathcal{D}\beta^* \|_2^2 + \mu_1 \| u_1 \|_1 + \mu_2 \| u_2 \|_1, \quad s.t., u_1 = A\beta^*, u_2 = B\beta^*, \tag{21}$$

where $\mathcal{D} = (\Phi_3^*, \Phi_4^*), \beta^* = (\beta_3^*, \beta_4^*)^T$, $A = (I, \mathbf{0})$, and $B = (\mathbf{0}, I)$. And $\mathbf{0}$ is zero matrix. I is identity matrix. Since the optimization Eq (21) is separable, w.r.t (β^*, u_1, u_2). The augmented Lagrangian of Eq (21) is

$$\begin{aligned}\mathcal{L}_{\tilde{\tau}_1,\tilde{\tau}_2}(\beta^*, u_1, u_2, b_1^*, b_2^*) &= \| R - \mathcal{D}\beta^* \|_2^2 + \mu_1 \| u_1 \|_1 + \mu_2 \| u_2 \|_1 \\ &+ \frac{\tilde{\tau}_1}{2} \| u_1 - A\beta^* + b_1^* \|_2^2 + \frac{\tilde{\tau}_2}{2} \| u_2 - B\beta^* + b_2^* \|_2^2,\end{aligned} \tag{22}$$

where b_1^*, b_2^* are the proper size Lagrangian multipliers. By ADMM, the minimizing of Eq (22) is solved by following three subproblems:

$$\beta^*\text{-}subproblem : \min_{\beta^*} \| R - \mathcal{D}\beta^* \|_2^2 + \frac{\tilde{\tau}_1}{2} \| u_1 - A\beta^* + b_1^* \|_2^2 + \frac{\tilde{\tau}_2}{2} \| u_2 - B\beta^* + b_2^* \|_2^2, \tag{23}$$

$$u_1\text{-}subproblem : \min_{u_1} \mu_1 \| u_1 \|_1 + \frac{\tilde{\tau}_1}{2} \| u_1 - A\beta^* + b_1^* \|_2^2, \tag{24}$$

$$u_2\text{-}subproblem : \min_{u_2} \mu_2 \| u_2 \|_1 + \frac{\tilde{\tau}_2}{2} \| u_2 - B\beta^* + b_2^* \|_2^2 . \tag{25}$$

We could use least squares method to solve β^*-*subproblem*:

$$\beta^* = (K_1)^{-1} q_1, \tag{26}$$

where $\beta^* \in \mathcal{R}^{2m\times 1}, K_1 = 2\mathcal{D}^T\mathcal{D} + \tilde{\tau}_1 A^T A + \tilde{\tau}_2 B^T B \in \mathcal{R}^{2m\times 2m}$ and $q_1 = 2\mathcal{D}^T R + \tilde{\tau}_1 A^T(u_1 + b_1^*) + \tilde{\tau}_2 B^T(u_2 + b_2^*) \in \mathcal{R}^{2m\times 1}$. The subproblem u_1 and u_2 have a closed form solution as mentioned in Algorithm 2. Thus, the solutions to the subproblems u_1 and u_2 are shown as following:

$$(u_1)_j = shrink((A\beta^*)_j - (b_1^*)_j, \frac{\mu_1}{\tilde{\tau}_1}), \tag{27}$$

$$(u_2)_j = shrink((B\beta^*)_j - (b_2^*)_j, \frac{\mu_2}{\tilde{\tau}_2}). \tag{28}$$

The main algorithm for iterative refinement is shown in Algorithm 3. We will illustrate the necessary of the Algorithm 3 in section 4.

Algorithm 3 (Iterative refinement)

Input: residual image R, μ_1, μ_2, $\tilde{\tau}_1$, $\tilde{\tau}_2$, $\Psi_3^*, \Psi_4^* \in \mathcal{R}^{n\times m}$

Output: coefficient β

1. **Initialization:** $\beta^{(1)} \leftarrow \mathbf{0}, \mu_1^{(1)} \leftarrow \mathbf{0}, \mu_2^{(1)} \leftarrow \mathbf{0}, b_1^{*(1)} \leftarrow \mathbf{0}, b_2^{*(1)} \leftarrow \mathbf{0}$
2. ***while not converged do***
3. $t \leftarrow t + 1$

4. $\beta^{*(t)} \leftarrow$ solve subproblem (23) for $u_1 = u_1^{(t-1)}, u_2 = u_2^{(t-1)}$, $b_1^* = b_1^{*(t-1)}, b_2^* = b_2^{*(t-1)}$,
5. $u_1^{(t)} \leftarrow$ solve subproblem (24) for $\beta^* = \beta^{*(t)}, b_1^* = b_1^{*(t-1)}$,
6. $u_2^{(t)} \leftarrow$ solve subproblem (25) for $\beta^* = \beta^{*(t)}, b_2^* = b_2^{*(t-1)}$,
7. $b_1^{*(t)} \leftarrow b_1^{*(t)} = b_1^{*(t-1)} + (u_1^{(t)} - A\beta^{*(t)})$,
8. $b_2^{*(t)} \leftarrow b_2^{*(t)} = b_2^{*(t-1)} + (u_2^{(t)} - A\beta^{*(t)})$.
9. ***end while***

3.3 Single image super-resolution based on approximated Heaviside functions and iterative refinement

Take different behaviours of Eqs (9) and (19) into consideration, we get the modified single super-resolution based on approximated Heaviside functions and iterative refinement, which is shown in Algorithm 4. The details of the proposed method could be found from the flow chart shown in the Fig 2.

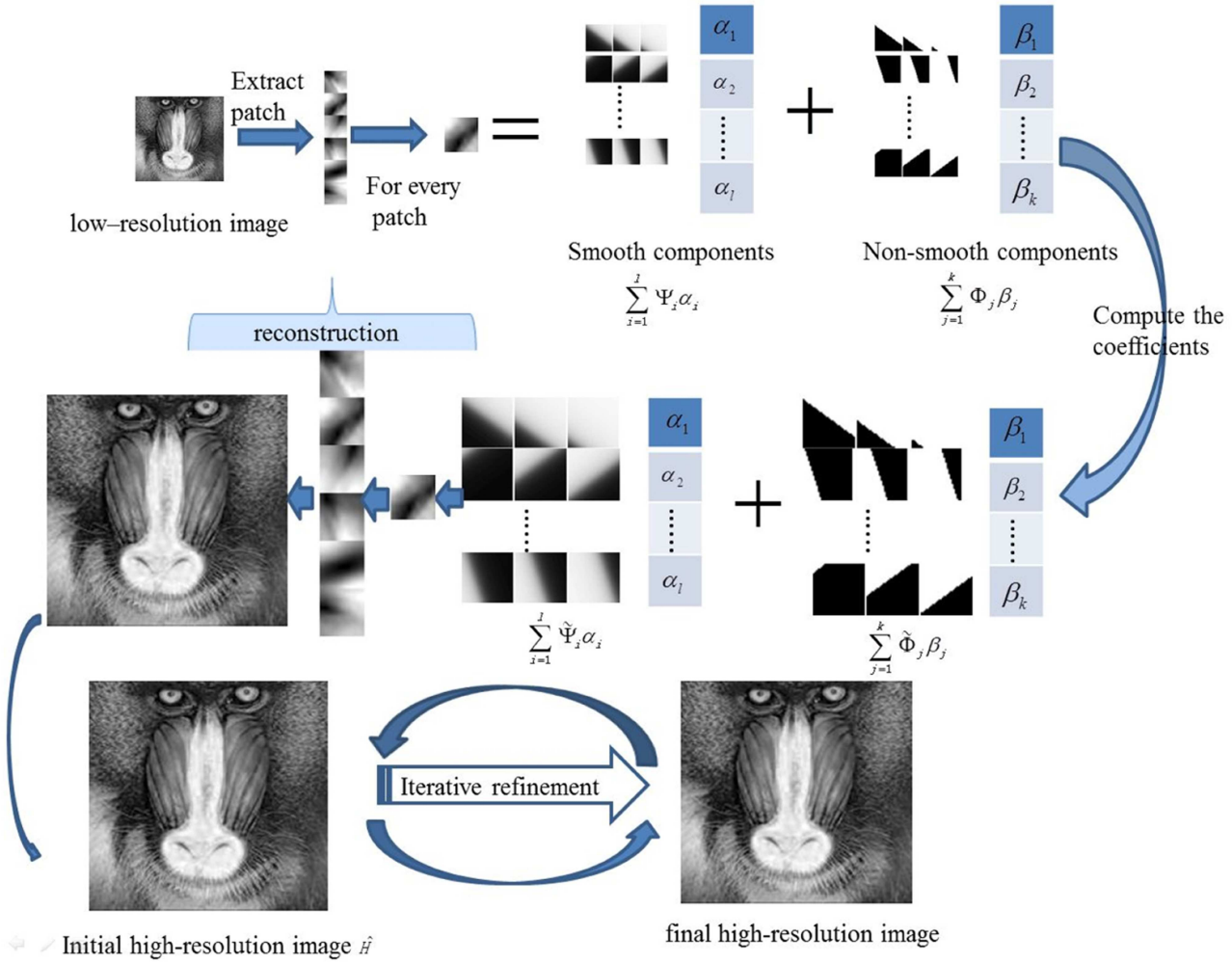

Fig 2. The flow chart of the proposed work.

https://doi.org/10.1371/journal.pone.0182240.g002

Algorithm 4

Input: Given the low-resolution image L (a vector form), λ_i, $\gamma_j > 0$ $(i = 1, 2, \cdots, l; j = 1, 2, \cdots, k)$, $\mu_1 > 0$, $\mu_2 > 0$, $\tilde{\tau}_1 > 0$, $\tilde{\tau}_2 > 0$. s: the upscaling factor. t: maximum iterations of the iterative refinement.
Output: high-resolution H

1. According to Eq (4), construct matrices $\Psi_i, \Phi_j \in \mathcal{R}^{n\times m}$, $\Phi_3^*, \Phi_4^* \in \mathcal{R}^{n\times m}$ on coarse grids, construct $\tilde{\Psi}_i, \tilde{\Phi}_j (i = 1, 2, \cdots, l; j = 1, 2, \cdots, k) \tilde{\Phi}_3^*, \tilde{\Phi}_4^* \in \mathcal{R}^{N\times m}$ on corresponding fine grids where $N = s^2 n$.

2. Compute the coefficients (α_i, β_j) according to Algorithm 2:

$$(\alpha_i, \beta_j) = \| L - \sum_{i=1}^{l} \Psi_i \alpha_i - \sum_{j=1}^{k} \Phi_j \beta_j \|_2^2 + \sum_{i=1}^{l} \lambda_i \| \alpha_i \|_2^2 + \sum_{j=1}^{k} \gamma_j \| \beta_j \|_1.$$

3. Get high-resolution image $\hat{H}$: $\hat{H} = \sum_{i=1}^{l} S_i + \sum_{j=1}^{k} E_j, S_i = \tilde{\Psi}_i \alpha_i, E_j = \tilde{\Phi}_j \beta_j$.

4. **Initialization:** $H^{(1)} = \hat{H}$.
for k = 1, 2, ⋯, t

   1. Compute the residual image $R^{(k)}$: $R^{(k)} = L - DH^{(k)}$.

   2. Compute the coefficients according to Algorithm 3:

$$(\beta_3^{*(k)}, \beta_4^{*(k)}) = \text{argmin} \| R^{(k)} - \Phi_3^* \beta_3^* - \Phi_4^* \beta_4^* \|_2^2 + \mu_1 \| \beta_3^* \|_1 + \mu_2 \| \beta_4^* \|_1.$$

   3. Update the high-resolution image:

$$H^{(k+1)} = H^{(k)} + E_1^{(k)} + E_2^{(k)},$$
$$\text{where } E_1^{(k)} = \tilde{\Phi}_3^* \beta_3^{*(k)}, E_2^{(k)} = \tilde{\Phi}_4^* \beta_4^{*(k)}.$$

end

5. Get non-smooth components: $E_1 = \sum_{i=1}^{t} E_1^{(i)}, E_2 = \sum_{i=1}^{t} E_2^{(i)}$.

6. Compute the final high-resolution image: $H = \hat{H} + conv(E_1, p) + conv(E_2, p)$, where $conv$ represents a convolution operator, p is a Gaussian kernel with a small size.

We use a Gaussian kernel p with a small size to make a convolution to avoid the oversharp information on the non-edge parts. And D is the bicubic downsampling operator.

However, the computation of the Algorithm 4 is very expensive, due to the large scale and non-sparse matrix $\mathcal{C}$ in Eq (12). For example, if a LR image is 512 × 512 and we choose 10 different directions, the size of matrix $\mathcal{C}$ will be equal to $(512^2 \times 10 \times 512^2)$. It is obviously large. Thus, it is observed that if we increase the number of smooth components and non-smooth components. The quality of an image usually gets improved at the cost of increased computation time and memory requirement. However, for large number of smooth components and non-smooth components, it is difficult to do SR on a limited hardware. We have experimentally seen that the use of more than $l = 1$, $k = 2$ in Algorithm 4 does not improve quality of the images significantly. In addition, $l = 1$, $k = 2$ work well on a regular desktop. Hence, we choose $l = 1$, $k = 2$ in our work. The optimization model can be simplified as:

$$\min_{\alpha_1, \beta_1, \beta_2} \| L - \Psi_1 \alpha_1 - \Phi_1 \beta_1 - \Phi_2 \beta_2 \|_2^2 + \lambda_1 \| \alpha_1 \|_2^2 + \lambda_2 \| \beta_1 \|_1 + \lambda_3 \| \beta_2 \|_1. \quad (29)$$

To simplify Eq (29), we set $\tilde{\mathcal{C}} = (\Psi_1, \Phi_1, \Phi_2)$, $\tilde{u} = (\alpha_1, \beta_1, \beta_2)$, $M = (\mathbf{I}, \mathbf{0}, \mathbf{0})$, $B_1^{'} = (\mathbf{0}, \mathbf{I}, \mathbf{0})$, $B_2^{'} = (\mathbf{0}, \mathbf{0}, \mathbf{I})$, where $\mathbf{0}$ is zero matrix, $\mathbf{I}$ is identity matrix. The Eq (29) can be rewritten as following.

$$\min_{\tilde{u}} \| L - \tilde{\mathcal{C}}\tilde{u} \|_2^2 + \lambda_1 \| M\tilde{u} \|_2^2 + \gamma_1 \| B_1^{'}\tilde{u}\|_1 + \gamma_2 \| B_2^{'}\tilde{u}\|_1. \tag{30}$$

Since l_1 term is not differentiable, we make two variable substitutions p_1, p_2 for $B_1^{'}\tilde{u}, B_2^{'}\tilde{u}$ to rewrite the Eq (30) as:

$$\begin{aligned} \min_{\tilde{u}} \| L - \tilde{\mathcal{C}}\tilde{u} \|_2^2 + \lambda_1 \| M\tilde{u} \|_2^2 + \gamma_1 \| p_1\|_1 + \gamma_2 \| p_2\|_1, \\ s.t., p_1 = B_1^{'}\tilde{u}, p_2 = B_2^{'}\tilde{u}. \end{aligned} \tag{31}$$

The Eq (31) is separable, w.r.t $(\tilde{u}, p_1, p_2)$. Here, we solve this problem by block-wise ADMM.

The augmented Lagrangian of Eq (31) is

$$\begin{aligned} \mathcal{L}(\tilde{u}, p_1, p_2, b_1, b_2) = \quad & \| L - \tilde{\mathcal{C}}\tilde{u} \|_2^2 + \lambda_1 \| M\tilde{u} \|_2^2 + \gamma_1 \| p_1\|_1 + \gamma_2 \| p_2\|_1 \\ & + \frac{\tau_1}{2} \| p_1 - B_1^{'}\tilde{u} + b_1 \|_2^2 + \frac{\tau_2}{2} \| p_2 - B_2^{'}\tilde{u} + b_2 \|_2^2, \end{aligned} \tag{32}$$

where b_1, b_2 are Lagrangian multipliers with proper size. The problem of minimizing $\mathcal{L}(\tilde{u}, p_1, p_2, b_1, b_2)$ could be solved by iteratively and alternatively by following subproblems:

$$\begin{aligned} \tilde{u}\text{-}subproblem \ : \ \min \| L - \tilde{\mathcal{C}}\tilde{u} \|_2^2 + \lambda_1 \| M\tilde{u} \|_2^2 + \frac{\tau_1}{2} \| p_1 - B_1^{'}\tilde{u} + b_1 \|_2^2 \\ + \frac{\tau_2}{2} \| p_2 - B_2^{'}\tilde{u} + b_2 \|_2^2, \end{aligned} \tag{33}$$

$$p_1\text{-}subproblem \ : \ \min \gamma_1 \| p_1\|_1 + \frac{\tau_1}{2} \| p_1 - B_1^{'}\tilde{u} + b_1 \|_2^2, \tag{34}$$

$$p_2\text{-}subproblem \ : \ \min \gamma_2 \| p_2\|_1 + \frac{\tau_2}{2} \| p_2 - B_2^{'}\tilde{u} + b_2 \|_2^2 . \tag{35}$$

The $\tilde{u}$-*subproblem* is a smooth quadratic problem. We solve it by least squares method as following:

$$\tilde{u} = K_1^{-1} q_1, \tag{36}$$

where

$$K_1 = \tilde{\mathcal{C}}^T\tilde{\mathcal{C}} + \lambda_1 M^T M + \frac{\tau_1}{2}(B_1^{'})^T B_1 + \frac{\tau_2}{2}(B_2^{'})^T B_2, \tag{37}$$

$$q_1 = \tilde{\mathcal{C}}^T L + \sum_{j=1}^{2} \frac{\tau_j}{2}(B_j^{'})^T (p_j + b_j). \tag{38}$$

The p_j-*subproblem*($j = 1, 2$) has a closed form solution Eqs (39) and (40) for each $(p_j)_i (j = 1, 2)$:

$$(p_1)_i = shrink((B_1'\tilde{u})_i - (b_1)_i, \frac{\gamma_1}{\tau_1}), \tag{39}$$

$$(p_2)_i = shrink((B_2'\tilde{u})_i - (b_2)_i, \frac{\gamma_2}{\tau_2}), \tag{40}$$

where $shrink(a, b) = \text{sign}(a)\max(|a| - b, 0)$.

We summarize $l = 1, k = 2$ in Algorithm 4 as modified single image based on approximated Heaviside functions and iterative refinement, which we call this algorithm as MAHFM, shown in Algorithm 5.

Algorithm 5 (MAHFM algorithm)

Input: Given the low-resolution image L (vector form), λ_1, γ_1, $\gamma_2 > 0$, $\mu_1 > 0$, $\mu_2 > 0$, $\tilde{\tau}_1 > 0$, $\tilde{\tau}_2 > 0$, s: the upscaling factor. t: maximum iterations of the iterative refinement.

Output: high-resolution H

1. According to Eq (4), construct matrices $\Psi_1, \Phi_1, \Phi_2, \Phi_3^*, \Phi_4^* \in \mathcal{R}^{n\times m}$ on coarse grids, construct $\tilde{\Psi}_1, \tilde{\Phi}_1, \tilde{\Phi}_2, \tilde{\Phi}_3^*, \tilde{\Phi}_4^* \in \mathcal{R}^{N\times m}$ on corresponding fine grids where $N = s^2 n$.
2. Compute the coefficients according to Algorithm 2:

$$(\alpha_1, \beta_1, \beta_2) = \text{argmin} \parallel L - \Psi_1\alpha_1 - \Phi_1\beta_1 - \Phi_2\beta_2 \parallel_2^2 + \lambda_1 \parallel \alpha_1 \parallel_2^2 + \gamma_1 \parallel \beta_1\parallel_1 + \gamma_2 \parallel \beta_2\parallel_1.$$

3. Get high-resolution image $\hat{H}$: $\hat{H} = S_1 + \sum_{j=1}^{2} E_j, S_1 = \tilde{\Psi}_1\alpha_1, E_j = \tilde{\Phi}_j\beta_j (j = 1, 2)$.
4. **Initialization:** $H^{(1)} = \hat{H}$.
   for $k = 1, 2, \cdots, t$
   1. Compute the residual image $R^{(k)}$: $R^{(k)} = L - DH^{(k)}$.
   2. Compute the coefficients according to Algorithm 3:

$$(\beta_3^{*(k)}, \beta_4^{*(k)}) = \text{argmin} \parallel R^{(k)} - \Phi_3^*\beta_3^* - \Phi_4^*\beta_4^* \parallel_2^2 + \mu_1 \parallel \beta_3^*\|_1 + \mu_2 \parallel \beta_4^*\|_1.$$

   3. Update the high-resolution image:

$$H^{(k+1)} = H^{(k)} + E_1^{(k)} + E_2^{(k)},$$
$$\text{where } E_1^{(k)} = \tilde{\Phi}_3^*\beta_3^{*(k)}, E_2^{(k)} = \tilde{\Phi}_4^*\beta_4^{*(k)}.$$

   end
5. Get non-smooth components: $E_1 = \sum_{i=1}^{t} E_1^{(i)}, E_2 = \sum_{i=1}^{t} E_2^{(i)}$.
6. Compute the final high-resolution image: $H = \hat{H} + conv(E_1, p) + conv(E_2, p)$, where *conv* represents a convolution operator, p is a Gaussian kernel with a small size.

Experiments

4.1 Numerical results

We have implemented our algorithm to some benchmark images whose high-resolution versions are available. For a quantitative comparison, we downscale actual HR images to their LR versions via bicubic interpolation and then generate HR images by MAHFM algorithm and other methods. For gray images, we perform the proposed algorithm to them directly. While

working for color images, the input image is first converted from RGB to YCbCr. We only perform the MAHFM algorithm on the illuminance channel because the human are more sensitive to the brightness information. The other two channels contain chromaticity information and they could be upsampled by bicubic interpolation. Finally, we convert three channels back to RGB to get the estimated color HR image.

We make the quantitative comparisons between the recovered HR images and the actual HR images. And we use root mean square error (RMSE) and peak signal to noise ratio (PSNR) on the illuminance channel to evaluate numerical performance.

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(h_i - \hat{h}_i)^2}, \tag{41}$$

$$\mathrm{PSNR} = 10 \cdot \log_{10}\left(\frac{(2^n - 1)^2}{\frac{1}{IJ}\sum_{i,j}(h_{ij} - \hat{h}_{ij})^2}\right), \tag{42}$$

where h, $\hat{h}$ are the vector-form of ground-truth image and the resulted high-resolution image, respectively. N represents testing times for one image. IJ is the original image consisting of $(I \times J)$ pixels. n is the number of bits per sample. The smaller RMSE is, the better performances of the method usually are. But PSNR has the opposite property. As mentioned in [7], if we apply MAHFM algorithm to one image, it would involve expensive computation. It follows that we could utilize image patches to reduce computation and storage significantly. In our work, we set patch size to be 6×6 and overlap to be 3.

First, we compare the MAHFM algorithm with AHFM algorithm [7], and the numerical results are shown in Table 1. The performance of MAHFM algorithm is compared with that of bicubic interpolation, a kernel regression method (denoted as "07'TIP" [34]), a fast upsampling method (denoted as "08'TOG" [22]), one state-of-the art learning-based method (denoted as "10'TIP" [12]), a fast image upsampling method via the displacement field (denoted as "14'TIP" [10]) and AHFM [7]. The numerical results by these methods are displayed in Table 2. Furthermore, we have also carried out our experiments for the upscaling factor $s = 3$. The quantitative measurements for them are shown in Table 3. As can be seen, these results reveal that the MAHFM algorithm is effective.

4.2 Visual comparisons

The test images (LR images) in Figs 3 and 4 are shown in Figs 3(a) and 4(a) of every figures, respectively. The image "baboon" and "peppers" are downloaded from http://decsai.ugr.es/cvg/dbimagenes/c512.php. The bottom of the web page has shown that these images are Copyright free. The version of the images in our paper are other special formats which are

Table 1. Comparison with AHFM algorithm and MAHFM algorithm(factor = 2).

Algorithms	PSNR/RMSE	*woodpecker*	*workman*	*flowers*	*butterfly*	*children*	*blackbutterfly*
AHFM	RMSE	3.3077	5.5733	10.3057	11.7942	3.7469	6.4146
	PSNR	38.4155	33.2085	27.8692	26.6974	036.6573	31.9874
MAHFM	RMSE	**3.0950**	**5.4714**	**6.7661**	**11.2689**	**3.5336**	**6.2337**
	PSNR	**38.6273**	**33.3689**	**31.5241**	**27.0932**	**37.1665**	**32.2359**

https://doi.org/10.1371/journal.pone.0182240.t001

Table 2. Quantitative results by different super-resolution algorithms (factor = 2).

Images	PSNR/RMSE	bicubic	07'TIP [34]	08'TOG [22]	10'TIP [12]	AHFM [7]	MAHFM
leaves	RMSE	15.6427	16.5913	16.1366	15.1104	14.8658	**14.8190**
	PSNR	24.2446	23.7332	23.9746	24.5433	24.6871	**24.7145**
father	RMSE	6.1870	8.5823	6.8589	5.6152	5.6024	**5.5610**
	PSNR	32.3012	29.4587	31.4057	33.1435	33.1633	**33.2354**
comic	RMSE	12.6995	16.4580	13.9571	11.1757	11.5335	**11.1000**
	PSNR	26.0551	23.8033	25.2349	27.1653	26.8916	**27.2243**
babyface	RMSE	4.6689	5.9649	5.1108	4.4681	4.4029	**4.3713**
	PSNR	34.7466	32.6188	33.9610	35.1284	35.2560	**35.3185**
lions	RMSE	7.3928	10.4452	8.2584	6.7658	6.4465	**6.2582**
	PSNR	30.7546	27.7524	29.7929	31.5244	31.9444	**32.2018**
tree	RMSE	13.2687	15.8486	13.8727	12.7048	12.4237	**12.3883**
	PSNR	25.6742	24.1310	25.2876	26.0515	26.2458	**26.2705**
peppers	RMSE	5.6703	7.1575	5.9815	5.1910	4.9485	**4.6897**
	PSNR	33.0587	31.0356	32.5945	33.8258	34.2413	**34.7080**

https://doi.org/10.1371/journal.pone.0182240.t002

converted by Photoshop from the sources above. We have shown the results on these images with different upscaling factors. For example, we have set the upscaling factor $s = 3$ in the Fig 3. The proposed algorithm is compared with bicubic interpolation, a learning-based method ("10'TIP" [12]), one state-of-the-art interpolation-based methods ("11'IPOL" [2]), one deep learning method("14'TIP" [10]) and the AHFM [7]. As can be seen, blur effect is generated by bicubic interpolation method. The learning-based method has comparable vision while we have to need extra training data to learn the relation between the test images and the training images. In addition, the interpolate-based methods ("11'IPOL") also show impressive performance. In particular, "14'TIP" shows superior vision on image edges and runs quite fast. But as shown in figures, not only does it lose small-scale texture but also introduces minimal jagged artificial and smooths non-edge regions. By contrast, the recovered HR images by the MAHFM algorithm are preserved sharpness on the edges, such as the texture of the mustache without ringing and blurring artifacts. Compared with the ground truth image, the MAHFM alorithm generates superior effect than others not only visually but also quantitatively.

Table 3. Quantitative results by different super-resolution algorithms (factor = 3).

Images	PSNR/RMSE	07'TIP [34]	08'TOG [22]	10'TIP [12]	14'TIP [10]	AHFM [7]	MAHFM
barbara	RMSE	15.7546	11.9587	12.2959	22.1094	12.0127	**11.8880**
	PSNR	24.1826	26.5771	26.3356	21.2393	26.1975	**26.6286**
view	RMSE	14.4619	**10.9190**	11.2941	21.9687	11.3169	11.2920
	PSNR	24.9263	**27.3672**	27.0731	21.2947	27.0562	27.0754
baboon	RMSE	20.3930	17.8625	17.8047	25.4575	18.1387	**17.7188**
	PSNR	21.9412	23.0920	23.1201	20.0145	23.1621	**23.1621**
fish	RMSE	18.4114	17.7298	17.5758	24.0806	17.3458	**17.2849**
	PSNR	22.8291	23.1567	23.2325	20.4975	23.3469	**23.3775**
tool	RMSE	8.5852	5.1631	5.2033	18.1161	4.8748	**4.8210**
	PSNR	29.4558	33.8726	33.8053	22.9695	34.3716	**34.4681**

https://doi.org/10.1371/journal.pone.0182240.t003

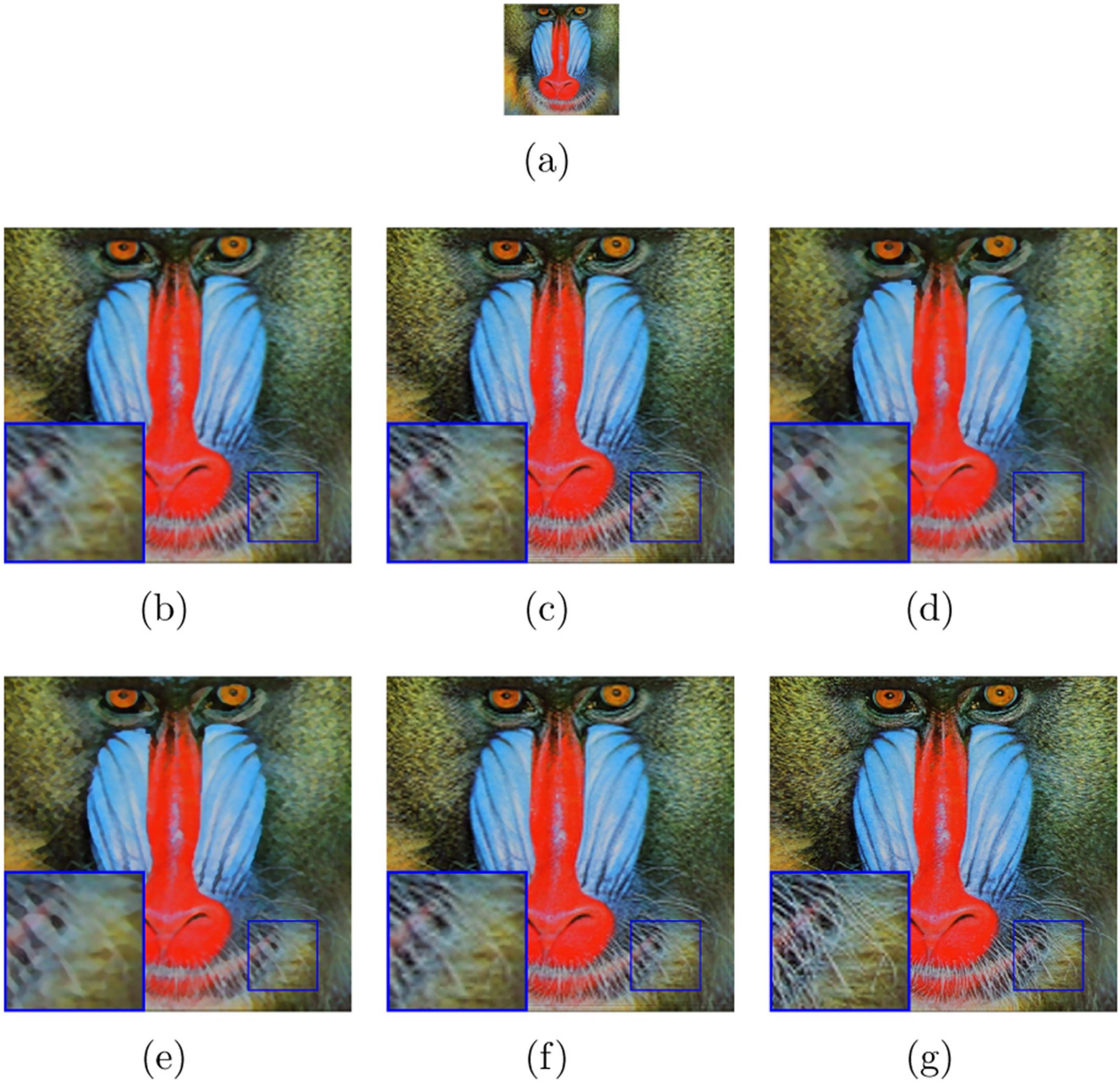

Fig 3. Results of "baboon" with the upscaling factor s = 3. (a) LR image, (b) bicubic, (c) 11'IPOL [2], (d) 14'TIP [10], (e) 10'TIP [12], (f) AHFM [7], (g) MAHFM.

https://doi.org/10.1371/journal.pone.0182240.g003

4.3 Comparisons between the AHFM and the proposed algorithm (MAHFM)

In this section we will illustrate the comparisons between the AHFM algorithm [7] and MAHFM algorithm. First, it is necessary to illustrate the difference between Algorithm 2 (let $l = 1$, $k = 2$) and AHFM algorithm. Although the only difference between the two methods is that the MAHFM algorithm is consisted of another term, which characters more sharp components. These could be seen much more clearly in Fig 5, which shows that the resulted HR image of Algorithm 2 (let $l = 1$, $k = 2$) is more sharp than that in the AHFM algorithm, especially at edges in Fig 5(c).

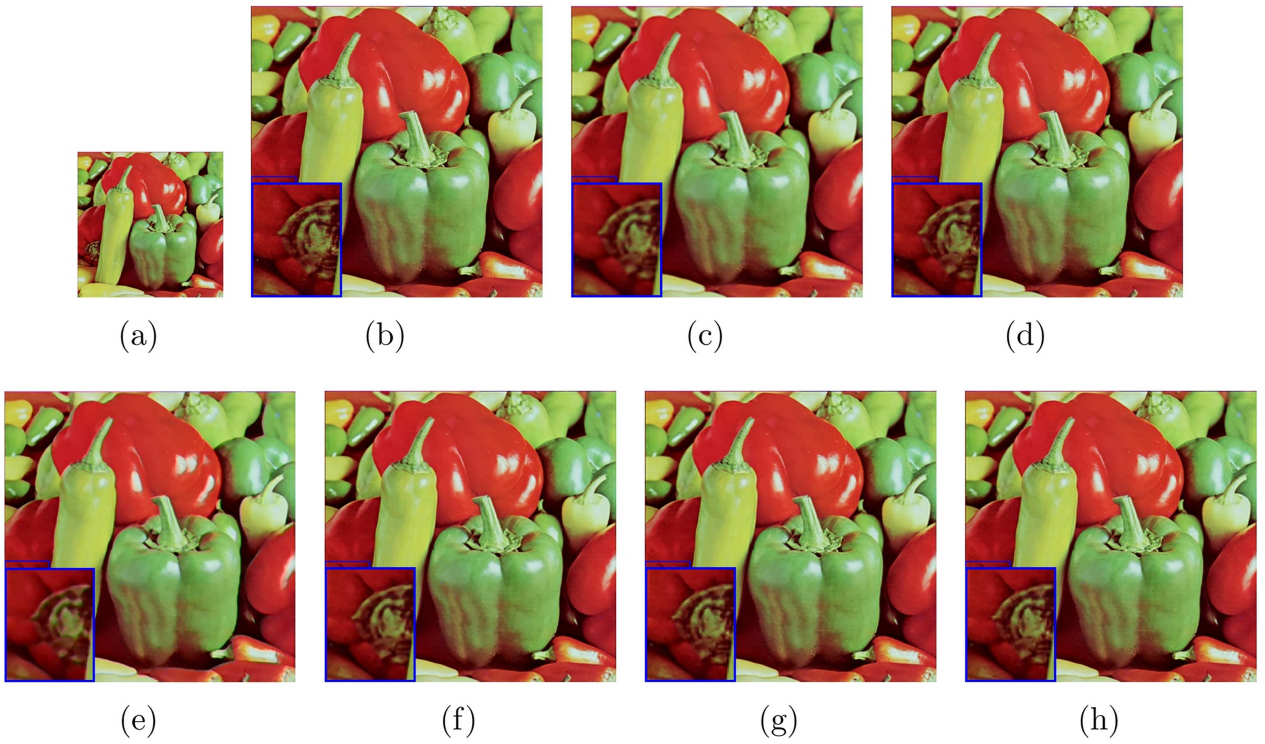

Fig 4. Results of "peppers" with the upscaling factor s = 2. (a) LR image, (b) ground truth, (c) bicubic, (d) 11'IPOL [2], (e) 14'TIP [10], (f)10'TIP [12], (g) AHFM [7], (h) MAHFM.

https://doi.org/10.1371/journal.pone.0182240.g004

Alternatively, we also compare the performance of the iterative refinement generated with that of the iterative AHFM algorithm [7]. To measure the effect of iterative refinement, we employ two classes of relative error defined as following:

$$\text{Reerror} = \frac{\| L - DH\|_F}{\| L\|_F}, \tag{43}$$

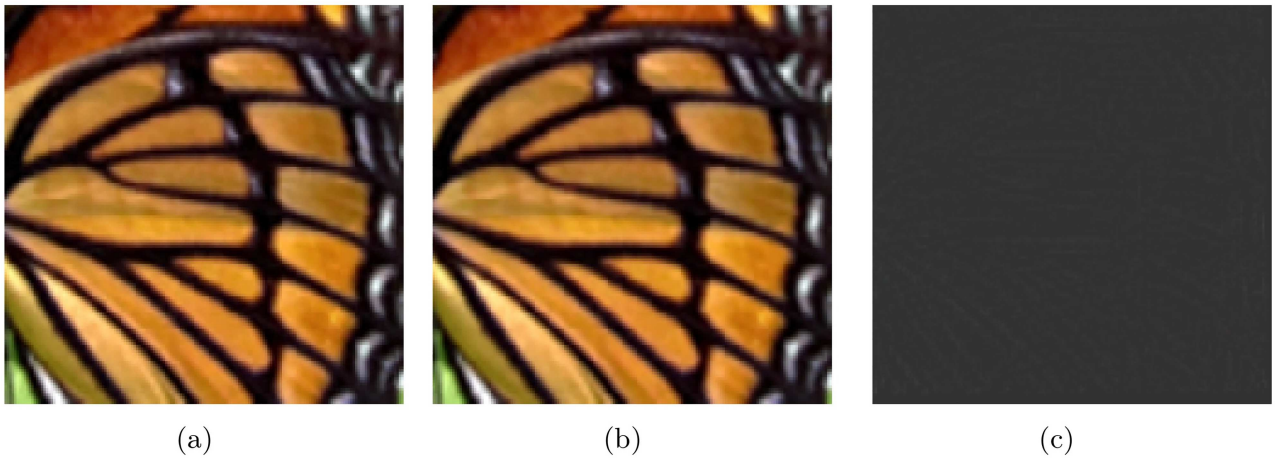

Fig 5. Comparisons between MAHFM and AHFM with factor *s* = 2. (a) MAHFM (RMSE = 8.7501, PSNR = 29.2905 dB). (b) AHFM [7] (RMSE = 9.0810, PSNR = 28.9681 dB). (c) residual (For better visualization, we add 0.5 to the intensities of the residual).

https://doi.org/10.1371/journal.pone.0182240.g005

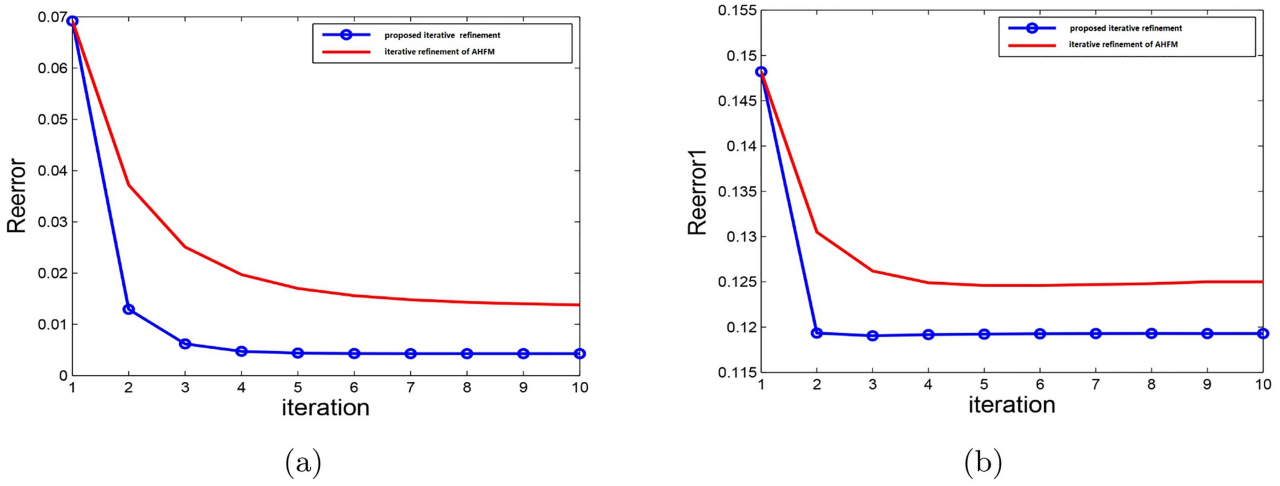

Fig 6. Comparisons between two iterative refinement methods. (a) Reerror. (b) Reerror1.

https://doi.org/10.1371/journal.pone.0182240.g006

and

$$\text{Reerror1} = \frac{\| H_{true} - H \|_F}{\| H_{true} \|_F}. \tag{44}$$

where L is a low resolution image. D is the downsampling operator. H is the last updated resulted image. H_{true} is the true high resolution image, and $\| \cdot \|_F$ is the *Frobenius* norm. After computing the relative errors of all the test images, we take the average of all the relative errors as the final results. The final results compared with the iterative AHFM algorithm in [7] are plotted in Fig 6, illustrating the Algorithm 3 could produce more details and less error than the original iterative refinement method. Besides, the precision of the proposed method is higher than that in original method. From the Fig 6, we can also get the iterative steps of the iterative refinement method. Only two iterations could be achieved the goal of refinement, which would also reduce the operation time.

4.4 Parameters

As mentioned in section 3, there are many parameters in the MAHFM algorithm, e.g. ξ_1, ξ_2, ξ_3, λ_1, λ_2, λ_3 and others. Although the parameters are so many, they are easy to select. In our work, we select these parameters mainly according to the experience. In particular, the parameters ξ_1, ξ_2, ξ_3 are especially important, because the sharpness for different components of an image are decided by them. First, we test ξ_1 by fixing ξ_2, ξ_3 and then tune ξ_1. In this way, we obtain a rough estimation of the three parameters (see Fig 7(a)). Note that there are some special points, which we can roughly estimate $\xi_1 \in [0.8, 1]$, $\xi_2 \in [10^{-2}, 10^{-1}]$ and $\xi_3 \in [10^{-4}, 10^{-2}]$. Fig 7(a) as well as reveals ξ_1 and ξ_3 are not sensitive to the selection of parameters. Thus we can select $\xi_1 = 0.8$ and $\xi_3 = 10^{-4}$ at first, and then tune ξ_2 from 10^{-2} to 10^{-1} by 10^{-2} time change, which is plotted in Fig 7(b). We could find that $\xi_2 = 6 \times 10^{-2}$ is the best fit. We conduct the same method to select other parameters. The final parameters of MAHFM algorithm are shown in Table 4.

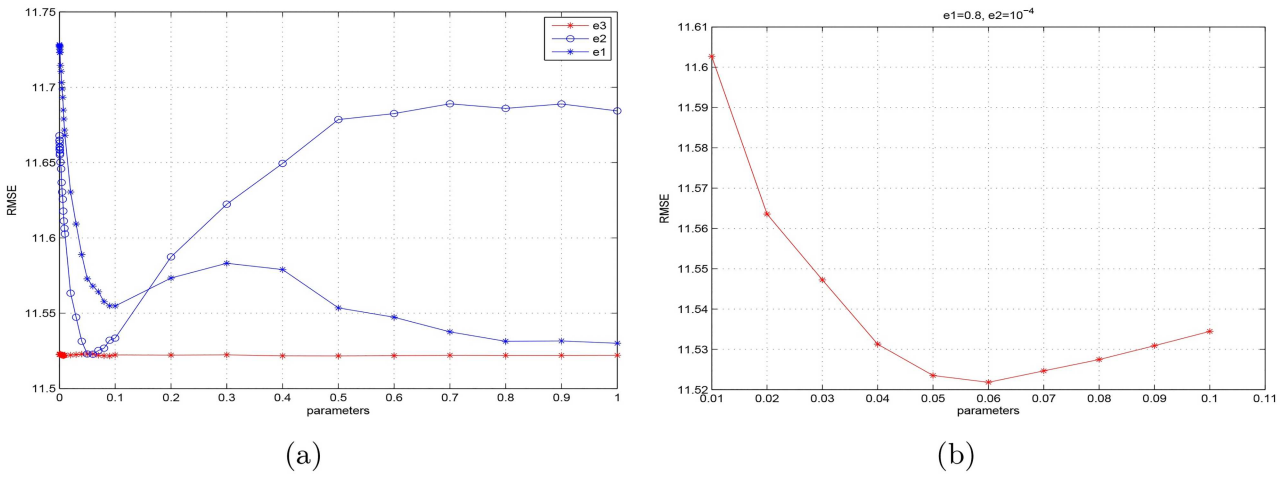

Fig 7. (a) The RMSE trend of the MAHFM algorithm as the different parameter varies when other parameters are roughly estimated (To simplify the experiment, e_1, e_2, e_3 represents ξ_1, ξ_2, ξ_3, respectively in this figure). (b) The RMSE trend of the MAHFM algorithm as the parameter ξ_2 varies, where $\xi_1 = 0.8$, $\xi_3 = 10^{-4}$.

https://doi.org/10.1371/journal.pone.0182240.g007

4.5 Computation time

From Figs 8 and 9, we can see the proposed method is a little slower in operation time. The main reason is that our algorithm is mainly to compute the coefficients u, however, due to many components in the proposed algorithm have to be character thereafter there are a lot of loop programs used Matlab. Hereto, to speed up the computation time, there is a lot of room. We believe that the limitations could be improved by using cmex in Matlab involving a lot of loops to speed up the code. From Fig 8, we can also find that Algorithm 3 runs more time than Algorithm 2, since it involves more matrix-vector representation. In addition, in our work, we also use the theory of patch to speed up. Based on this, we can also consider utilizing the parallel computing.

4.6 The differences between Heaviside function and wavelets

In the image processing field, there exist some other basis functions such as wavelets [35]. Therefore, we discuss the difference between the Heaviside function and wavelets. To test the difference, we write a program about the representation of super resolution based on the haar wavelet. The quantitative comparisons are shown in Table 5. Compared Fig 10 with Fig 4, we can see that the algorithm based on Heaviside function has got more details about the non-smooth components than the wavelets do. The algorithm based on wavelets mainly splits one image into different sub bands. The first layer scale coefficients and wavelet coefficients of the image are extracted from the structure of wavelet decomposition. The different sub bands focus on the different ways, such as the information about horizontal, vertical and other

Table 4. Parameters of MAHFM algorithm.

ξ_1	ξ_2	ξ_3	λ_1	γ_1	γ_2	τ_1
1	6×10^{-2}	1×10^{-4}	5×10^{-4}	1×10^{-3}	1×10^{-5}	3×10^{-3}
τ_2	$\tilde{\xi}_1$	$\tilde{\xi}_2$	μ_1	μ_2	$\tilde{\tau}_1$	$\tilde{\tau}_2$
1×10^{-4}	1×10^{-4}	1×10^{-5}	1×10^{-4}	5×10^{-3}	1×10^{-5}	1×10^{-6}

https://doi.org/10.1371/journal.pone.0182240.t004

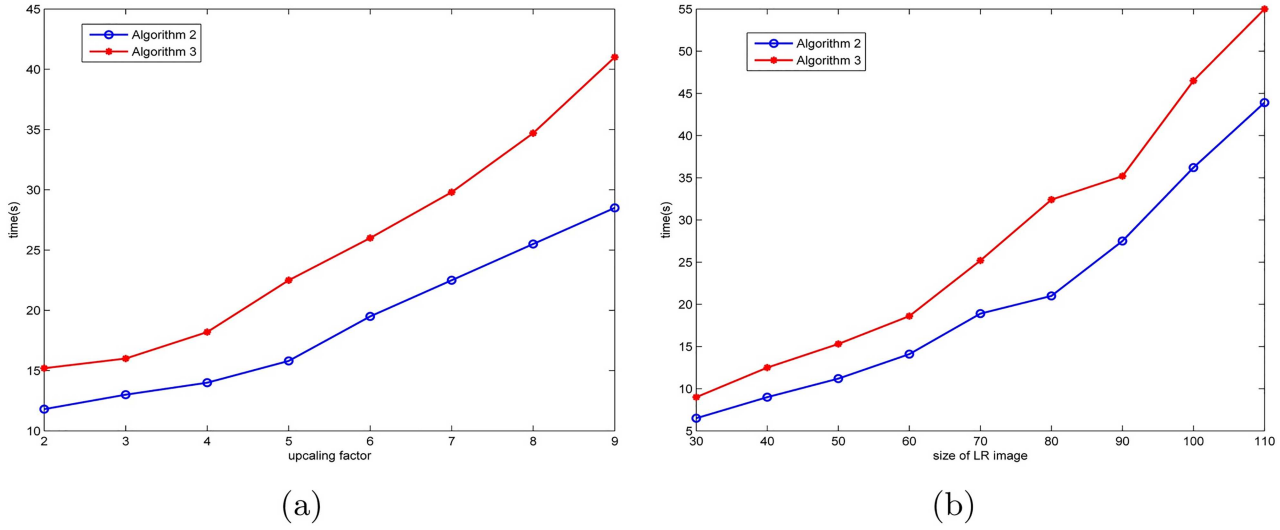

Fig 8. (a) Computation time of Algorithm 2 ($l = 1$, $k = 2$) and Algorithm 3 vs. upscaling factor for the low-resolution image "butterfly" with LR size 60×60. (b) Computation time of Algorithm 2 ($l = 1$, $k = 2$) and Algorithm 3 vs. the size of low-resolution image "butterfly", the size of low-resolution image is increased from 30×30 to 110×110 and the upscaling factor is always set to be 4. Note that, to reduce the instability of Matlab, the computation time is the average of 10 runs.

https://doi.org/10.1371/journal.pone.0182240.g008

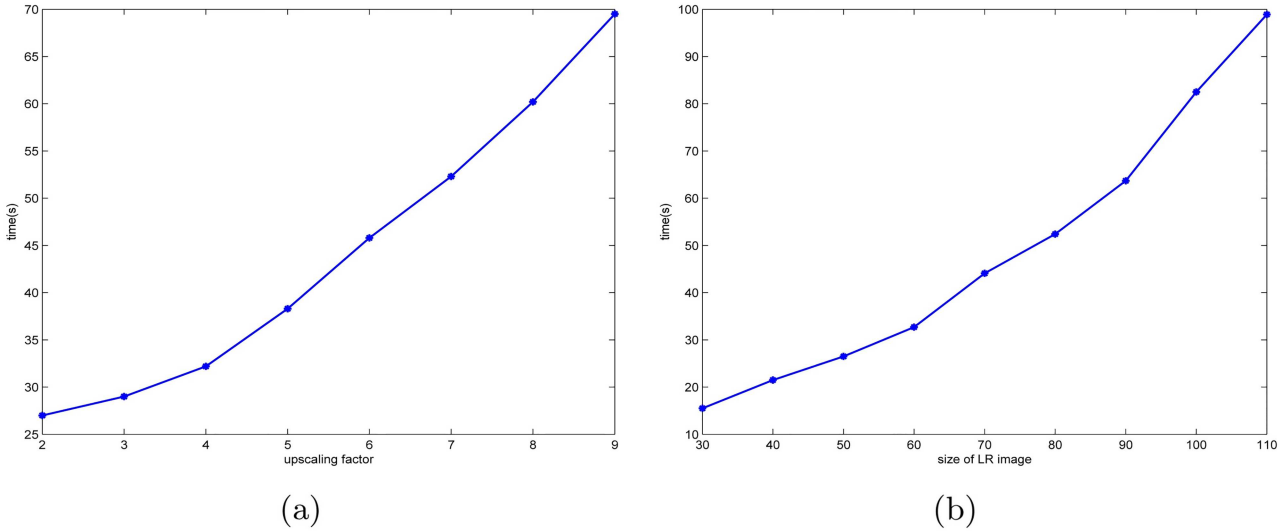

Fig 9. (a) Computation time of MAHFM (i.e., Algorithm 5) vs. upscaling factors for the low-resolution image "butterfly" with LR size 60×60. (b) Computation time of MAHFM vs. the size of low-resolution image "butterfly", the size of low-resolution image is increased from 30×30 to 110×110 and the upscaling factor is always set to be 4. Note that, at the common point with Fig 8 (i.e., upscaling factor 4 and 60×60 LR size), it has slightly different computation due to the instability of Matlab, thus we present the computation time by the average of 10 runs to reduce the gap.

https://doi.org/10.1371/journal.pone.0182240.g009

Table 5. Comparison with super resolution based on the basis of wavelets and Heaviside function.

Algorithms	RMSE/PSNR	*peppers* ($s = 2$)	*babyface* ($s = 2$)	*barbara* ($s = 3$)	*girl* ($s = 3$)	*baboon* ($s = 3$)
wavelets	RMSE	6.1932	5.1663	13.0254	6.2360	18.5015
	PSNR	32.2925	33.9518	25.8350	32.2327	22.7867
Heaviside function	RMSE	**4.6897**	**4.3713**	**11.8880**	**4.8210**	**17.7188**
	PSNR	**34.7080**	**35.3185**	**26.6286**	**34.4681**	**23.1621**

https://doi.org/10.1371/journal.pone.0182240.t005

Fig 10. Results of "peppers" with the upscaling factor s = 2 via the representation of Haar wavelets.

https://doi.org/10.1371/journal.pone.0182240.g010

directions. Every sub band persists the information only about the corresponding direction. If some operators, such as bicubic interpolation, are applied, error would be generated. And then the error would make pixels overlap, causing Gibbs effect. Differently, the basis is generated by the corresponding Heaviside functions and the representation coefficients are extracted from the low resolution image. The MAHFM algorithm could represent different sharp components via different classes of Heaviside functions as well as possible, which makes the information minimize the loss.

Conclusion

In this paper, we have proposed a general framework of approximated Heaviside functions model that can describe different smooth components and non-smooth components in an image. The different components of an image could be approximately represented by different classes of approximated Heaviside functions (AHFs). With only one LR image input, we could compute the representation coefficients of AHFs, and then utilized these representation coefficients to obtain the high-resolution images. In addition, to pick up more image details, we employed an iterative refinement algorithm according to the residuals between the LR input and the downsampled LR image. To reduce computation and storage size, we applied the MAHFM algorithm to image patches. In particular, we discussed how to choose parameters. Extensive examples have been provided in the experiments to show the effectiveness of the MAHFM algorithm both quantitatively and visually.

Author Contributions

Conceptualization: Xin-Yu Wang.

Formal analysis: Xin-Yu Wang, Ting-Zhu Huang, Liang-Jian Deng.

Methodology: Xin-Yu Wang, Ting-Zhu Huang, Liang-Jian Deng.

Software: Xin-Yu Wang.

Supervision: Ting-Zhu Huang, Liang-Jian Deng.

Validation: Xin-Yu Wang.

Visualization: Xin-Yu Wang.

Writing – original draft: Xin-Yu Wang.

Writing – review & editing: Xin-Yu Wang.

References

1. Elad M, Feuer A. Restoration of a single superresolution image from several blurred, noisy, and undersampled measured images. IEEE Transactions on Image Processing. 1997; 6:1646–1658. https://doi.org/10.1109/83.650118 PMID: 18285235
2. Getreuer P. Image interpolation with contour stencils. Image Processing On Line. 2011; 1. https://doi.org/10.5201/ipol.2011.g_igcs
3. Mueller N, Lu Y, Do M N. Image interpolation using multiscale geometric representations. Proc Spie. 1973; 13:341–353.
4. Zhang L, Wu X. An edge-guided image interpolation algorithm via directional filtering and data fusion. IEEE Transactions on Image Processing, 2006; 15:2226–2238. https://doi.org/10.1109/TIP.2006.877407 PMID: 16900678
5. Li X, Orchard M T. New edge-directed interpolation. IEEE Transactions on Image Processing. 2001; 10:1521–1527. https://doi.org/10.1109/83.951537 PMID: 18255495
6. Getreuer P. Contour stencils: Total Variation along curves for adaptive image interpolation. SIAM Journal on Imaging Sciences. 2011; 4:954–979. https://doi.org/10.1137/100802785
7. Deng L J, Guo W H, Huang T Z. Single image super-resolution by approximated Heaviside functions. Information Sciences. 2016; 348:107–123. https://doi.org/10.1016/j.ins.2016.02.015
8. Cha Y, Lee G Y, Kim S. Image zooming by curvature interpolation and iterative refinement. SIAM Journal on Imaging Sciences. 2014; 7:1284–1308. https://doi.org/10.1137/130907057
9. Kim H, Cha Y, Kim S. Curvature interpolation method for image zooming. IEEE Transactions on Image Processing. 2011; 20:1895–1903. https://doi.org/10.1109/TIP.2011.2107523 PMID: 21257378
10. Wang L F, Wu H Y, Pan C H. Fast image upsampling via the displacement field. IEEE Transactions on Image Processing. 2014; 23:5123–5135. https://doi.org/10.1109/TIP.2014.2360459 PMID: 25265631

11. Pulak P, Pal. N. R, Chanda B. A fuzzy-rule-based approach for single frame super resolution. IEEE Transactions on Image Processing. 2014; 23:2277–2290. https://doi.org/10.1109/TIP.2014.2312289
12. Yang J, Wright J, Huang T S, Ma Y. Image super-resolution via sparse representation. IEEE Transactions on Image Processing. 2010; 19:2861–2873. https://doi.org/10.1109/TIP.2010.2050625 PMID: 20483687
13. Dong C, Chen C L, He K, Tang X. Learning a deep convolutional network for image super-resolution. European Conference on Computer Vision(ECCV). 2014;184–199.
14. Peleg T, Elad M. A statistical prediction model based on sparse representations for single image super-resolution. IEEE Transactions on Image Processing. 2014; 23:2569–2582. https://doi.org/10.1109/TIP.2014.2305844 PMID: 24815620
15. Takagi T, Sugeno M. Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. on Syst. man and Cybern. 1985; 15:387–403.
16. Farsiu S, Robinson M D, Elad M. Fast and robust multiframe super resolution. IEEE Transactions on Image. 2004; 13:1327–1344. https://doi.org/10.1109/TIP.2004.834669
17. Fattal R. Image upsampling via imposed edge statistics. Acm Transactions on Graphics. 2007; 26:95. https://doi.org/10.1145/1276377.1276496
18. Glasner D, Bagon S, Irani M. Super-resolution from a single image. IEEE International Conference on Computer Vision. 2009;349–356.
19. Chambolle A, Pock T. A first-order primal-dual algorithm for convex problems with applications to imaging. Journal of Mathematical Imaging and Vision. 2011; 40:120–145. https://doi.org/10.1007/s10851-010-0251-1
20. Irani M, Peleg S. Super resolution from image sequence. International Conference on Pattern Recognition. 1990;2:115–120.
21. Komatsu T, Igarashi T, Aizawa K. Very high resolution imaging scheme with multiple different-aperture cameras. Signal Processing Image Communication. 1993; 5:511–526. https://doi.org/10.1016/0923-5965(93)90014-K
22. Shan Q, Li Z R, Jia J Y, Tang C K. Fast image/video upsampling. ACM Transactions on Graphics (TOG). 2008; 27. https://doi.org/10.1145/1409060.1409106
23. Fernandez G C, Candès E J. Super-resolution via transform-invariant group-sparse regularization. IEEE International Conference on Computer Vision. 2013;3336–3343.
24. Liang-Jian Deng, Weihong Guo, Ting-Zhu Huang. Single image super-resolution via an iterative reproducing kernel Hilbert space method. IEEE Transactions on Circuits and Systems for Video Technology. 2016; 26(11):2001–2014. https://doi.org/10.1109/TCSVT.2015.2475895
25. Liang-Jian Deng, Huiqing Guo, Ting-Zhu Huang. A fast image recovery algorithm based on splitting deblurring and denoising. Journal of Computational and Applied Mathematics, 2015; 287:88–97. https://doi.org/10.1016/j.cam.2015.03.035
26. He B, Tao M, Yuan X. Block-wise alternating direction method of multipliers for multiple-block convex programming and beyond. SMAI Journal of Computational Mathematics. 2015; 1:145–174. https://doi.org/10.5802/smai-jcm.6
27. Kainen P C, Kůrková V V., Vogt A. Best approximation by linear combinations of characteristic functions of half-space. Journal of Approximation Theory. 2003; 122:151–159. https://doi.org/10.1016/S0021-9045(03)00072-8
28. Zhao X L, Wang F, Ng M K. A new convex optimization model for multiplicative noise and blur removal. SIAM Journal on Imaging Sciences. 2014; 7:456–475. https://doi.org/10.1137/13092472X
29. Xu Y, Huang T Z, Liu J. An augmented Lagrangian algorithm for total bounded variation regularization based image deblurring. Journal of the Franklin Institute. 2014; 351:3053–3067. https://doi.org/10.1016/j.jfranklin.2014.02.009
30. Liu G, Huang T Z, Liu J. High-order TV L1-based images restoration and spatially adapted regularization parameter selection. Computers and Mathematics with Applications. 2014; 67:2015–2026. https://doi.org/10.1016/j.camwa.2014.04.008
31. Chen C, Ng M K, Zhao X L. Alternating direction method of multipliers for nonlinear image restoration problems. IEEE Transactions on Image Processing. 2015; 24:33–43. https://doi.org/10.1109/TIP.2014.2369953 PMID: 25398177
32. Zhang Y D, Peterson B S, Ji G L, Dong Z C. Energy Preserved Sampling for Compressed Sensing MRI. Computational and Mathematical Methods in Medicine. 2014; 5:546814, 12 pages.
33. Wang Y, Yang J, Yin W, Zhang Y. A new alternating minimization algorithm for total variation image reconstruction. SIAM Journal on Imaging Sciences. 2008; 1:248–272. https://doi.org/10.1137/080724265

34. Takeda H, Farsiu S, Milanfar P. Kernel regression for image processing and reconstruction. IEEE Transactions on Image Processing. 2007; 16:349–366. https://doi.org/10.1109/TIP.2006.888330 PMID: 17269630

35. Zhang Y D, Wang S H, Ji G L, Dong Z C. Exponential wavelet iterative shrinkage thresholding algorithm with random shift for compressed sensing magnetic resonance imaging. Ieej Transactions on Electrical and Electronic Engineering. 2015; 1:115–132.