Journal Pre-proof

Two-direction Self-learning Super-Resolution Propagation Based on Neighbor Embedding

Jian Xu , Yan Gao , Jun Xing, Jiulun Fan, Qiannan Gao, Shaojie Tang

PII: S0165-1684(21)00072-4
DOI: https://doi.org/10.1016/j.sigpro.2021.108033
Reference: SIGPRO 108033

To appear in: *Signal Processing*

Received date: 12 August 2020
Revised date: 22 January 2021
Accepted date: 7 February 2021

Please cite this article as: Jian Xu , Yan Gao , Jun Xing, Jiulun Fan, Qiannan Gao, Shaojie Tang, Two-direction Self-learning Super-Resolution Propagation Based on Neighbor Embedding, *Signal Processing* (2021), doi: https://doi.org/10.1016/j.sigpro.2021.108033

- We found the vertical similarity prior in the image pyramid.
- We proposed strategy with random oscillation, horizontal propagation, and vertical propagation to update the matching patches.
- We proposed a coarse-to-fine feature selection method for patch matching.

Two-direction Self-learning Super-Resolution Propagation Based on Neighbor Embedding ☆

Jian Xu, Yan Gao, Jun Xing, Jiulun Fan, Qiannan Gao, Shaojie Tang

School of Communications and Information Engineering, Xi'an University of Posts and Telecommunications, Xi'an, 710121, China

Key Laboratory of Electronic Information Application Technology for Scene Investigation, Ministry of Public Security, Xi'an, 710121, China

Abstract

Neighbor embedding (NE) is a widely used super-resolution (SR) algorithm, but the one-to-many problem always degrades the performance of NE. The simplest way to avoid this performance degradation is to extract image features from low-resolution (LR) patches, that correctly reflect the features in the corresponding high-resolution (HR) patches. In this paper, we propose several feature extraction methods to extract patch features in LR space, and use coarse-to-fine patch matching methods to select matching patches for each test patch and find the best matching patch candidate to update the matching patch. Since finding matching patches using a training set is exhaustive work in NE, we explore the traditional local/non-local similarity prior and propose vertical similarity in the image pyramid to accelerate the matching patch search process. To accomplish this, we propose a random oscillation+horizontal propagation+vertical propagation strategy to update the matching patches. The experimental results show that the proposed method is superior to many existing self-learning-based methods, but it is inferior to many external-learning-based methods. These results show that the proposed feature extraction method and oscillation propagation method are useful for finding proper matching patches in NE.

Keywords: super-resolution, random oscillation, neighbor-embedding, self-learning, principal component analysis

☆Corresponding author
Email address: jiulun_fan@126.com (Jiulun Fan)

1. Introduction

Single image super-resolution (SISR) generates a high-resolution (HR) image according to one or a group of low-resolution (LR) images[1, 2]. The most difficult task of SISR is to generate HR details according to the information that is provided by the LR images, because it is not easy to estimate the degradation parameters[3]. If we want to recover precise details, machine learning technologies provide many tools to learn the relationship between LR and HR images[4]. These tools include neural networks (such as convolutional neural networks (CNN)[5, 6, 7, 8, 9, 10], recurrent neural networks (RNN)[11], deep residual networks (ResNet)[12, 13, 14, 15, 16], and generative adversarial networks (GAN)[17, 18, 19, 20]), and sparse representation dictionaries[21, 22, 23, 24, 25, 26, 27]. Thus, a group of training images should be collected, and a training set that contains corresponding LR and HR images should be produced.

One machine learning algorithm is called 'self-learning'[28, 29, 30]. This algorithm generates corresponding LR and HR training sets by using only the input LR test image itself. Compared with 'external-learning' algorithms that collect training sets using other images[31, 8, 32, 33, 34, 35], there are several advantages to self-learning.

(1) Self-learning combines the training and testing processes and can adapt to different magnification factors, because the corresponding LR and HR training data sets can be generated according to the magnification factors[36]. However, an external learning algorithm separates the training and testing processes. When the parameters of machine learning tools are determined, the magnification factor can not change.

(2) Self-learning can generate high quality training images through a pyramid structure[28, 37, 38, 39]. Several related works[40, 41, 42] have shown that external learning recovers more diverse details with sparser distributions than internal learning, but internal learning concentrates on recovering specific details with a denser distribution. Therefore, self-learning can obtain good results on images with many repeated structures.

(3) Self-learning can be used in some applications without enough training images[43]. For example, some doctors can only obtain a very small number of images for lesion locations.

(4) Since self-learning only generates a small number of training samples

from the LR test image itself, it does not require high-end computational and storage equipment in most cases.

However, self-learning still has many problems.

(1) Since only a small number of training samples can be generated using the LR test image itself, some machine learning tools such as the CNN[44] and sparse representation dictionary[24] do not work well. Affine transformations[45, 28](such as scaling, flipping, rotating and cropping) are always applied to the input LR test images to generate more training samples[46], but the number of training samples is still not enough for many machine learning tools.

(2) For each test image, the self-learning process contains training and testing processes. When seeking to accelerate the SR process, some high computational complexity training algorithms can not be used for self-learning.

To solve the above problems and make good use of their advantages, we propose a self-learning algorithm. The proposed algorithm utilizes a random oscillation to generate matching patch candidates, and then selects the proper features for vertical and horizontal propagations. This algorithm is based on neighbor embedding (NE)[47, 48]. The NE finds similar LR training patches for each test patch and then uses the corresponding HR training patches to reconstruct the HR test image. The reasons for using NE are as follows. First, the performance of NE is highly related to the training set. If the training set is similar to the testing set, NE achieves good performance. Because of the self-similarity prior[49, 21], the training set that is generated by the pyramid structure[50] is similar to the testing set, especially when the LR testing image has repetitive structures. Second, several strategies can be used with the NE algorithm to accelerate the patch matching process.

However, there are two problems that should be solved in the NE algorithm. (1) Since NE works on image patches, proper patch features should be extracted to find proper matching patch. (2) It has high time costs when finding the matching patches for each test patch. To solve the problems above, the contributions of the proposed algorithm are as follows. (1) We use the local binary pattern (LBP) feature to perform coarse selection for patch matching and select the best patch candidate to update the matching patch. The dimension of the LBP feature is the same as the variance but it can better reflect the textures of image patches. (2) We propose a random oscillation+horizontal propagation+vertical propagation algorithm to find the candidate matching patches. By using both the local/non-local similarity prior and the vertical similarity prior, the proposed algorithm can

improve the quality of the reconstruction result.

The remainder of this paper is organized as follows: Section 2 reviews the related work. Section 3 describes the proposed algorithm in detail. Section 4 presents the experimental results. Section 5 concludes this paper.

2. Related work

The main purpose of self-learning-based SR is to use a machine learning method to magnify an LR image without relying on external data set. There are three key problems that should be solved in self-learning SR based on NE. The first is how to build the training data set. The second is how to extract the features of patches. The third is how to find the best matching patch. We will illustrate the methods for dealing with these problems in the following. We define the variables in Table 1.

Table 1: Variables Definition.

Variables	Definition
$\boldsymbol{\Psi}_L, \boldsymbol{\Psi}_H$	The LR and HR image pyramids
$\hat{\boldsymbol{\Psi}}_L, \boldsymbol{\Psi}_{lbp}$	The PCA and LBP feature pyramids
$\boldsymbol{I}_L^{(\varphi)}, \boldsymbol{I}_H^{(\varphi)}$	The φth level LR and HR images in $\boldsymbol{\Psi}_L$ and $\boldsymbol{\Psi}_H$
$\hat{\boldsymbol{I}}_L^{(\varphi)}, \breve{\boldsymbol{I}}_L^{(\varphi)}$	The φth layer PCA and LBP feature pyramids in $\hat{\boldsymbol{\Psi}}_L$ and $\boldsymbol{\Psi}_{lbp}$
$\boldsymbol{E}$	The PCA dictionary
$\boldsymbol{e}^{(\delta)}$	The δth PCA dictionary atom
d	The number of atoms
$\boldsymbol{I}_L$	The input LR image
$\widehat{\boldsymbol{I}}_H$	The reconstructed HR image
$\hat{\boldsymbol{I}}_L^{p,(\varphi)}$	The feature of pth layer in the φth layer image of the PCA feature pyramid $\hat{\boldsymbol{I}}_L^{(\varphi)}$
$\boldsymbol{P}_t^{(k)}, \boldsymbol{P}_s^{(k)}$	The test patch and its matching patch
$\boldsymbol{P}_s^{(k,N)}$	The matching patch in the Nth iteration of patch $\boldsymbol{P}_t^{(k)}$
$\tilde{\boldsymbol{P}}$	The candidate patches of matching patch
$\boldsymbol{P}_s^{(k,N+1)}$	The best candidate patch from candidate patches $\tilde{\boldsymbol{P}}$
$\tilde{\boldsymbol{P}}_s^{(k,N)}$	The matching patch of $\tilde{\boldsymbol{P}}_t^{(k)}$ in the Nth iteration
$\tilde{\tilde{\boldsymbol{P}}}_s^{(k,N)}$	The adjacent patch of $\boldsymbol{P}_s^{(k,N)}$
μ_c , μ_q	The center pixel and its surrounding pixel
(x_c, y_c)	The coordinates of μ_c
$\boldsymbol{P}_s^{(k,N+1)}, \boldsymbol{P}_s^{(\mu,N+1)}$	The best matching patch of $\boldsymbol{P}_t^{(k)}$ and $\boldsymbol{P}_t^{(\mu)}$ in the $(N+1)$th iteration
$(x_s^{(k,N+1)}, y_s^{(k,N+1)})$	The position coordinates of $\boldsymbol{P}_s^{(k,N+1)}$
$\boldsymbol{P}_t^{(\mu)}$	The test patch whose coordinates are same as $\boldsymbol{P}_s^{(k,N+1)}$
$\boldsymbol{\alpha}_s^{(k,N+1)}, \boldsymbol{\alpha}_t^{(k)}$	The PCA feature of patch $\boldsymbol{P}_s^{(k,N+1)}$ and $\boldsymbol{P}_t^{(k)}$

2.1. Generate dataset

Machine learning methods always rely on training data sets. Because self-learning-based SR only relies on a single input LR image, we must generate enough training data samples.

The LR and HR image pyramids are generated by the input LR image $\boldsymbol{I}_L$. The first layer of the HR image pyramids is $\boldsymbol{I}_H^{(0)} = \boldsymbol{I}_L$. $\boldsymbol{I}_H^{(\varphi)}$ is generated by down-sampling $\boldsymbol{I}_H^{(\varphi-1)}$. The magnification factor between the adjacent layers is s. The image layers in the LR image pyramid $\boldsymbol{\Psi}_L$ are generated by up-sampling the image layers in the HR image pyramid $\boldsymbol{\Psi}_H$. $\boldsymbol{I}_L^{(\varphi)}$ is generated by up-sampling $\boldsymbol{I}_H^{(\varphi+1)}$. The up-sampling factor is $1/s$.

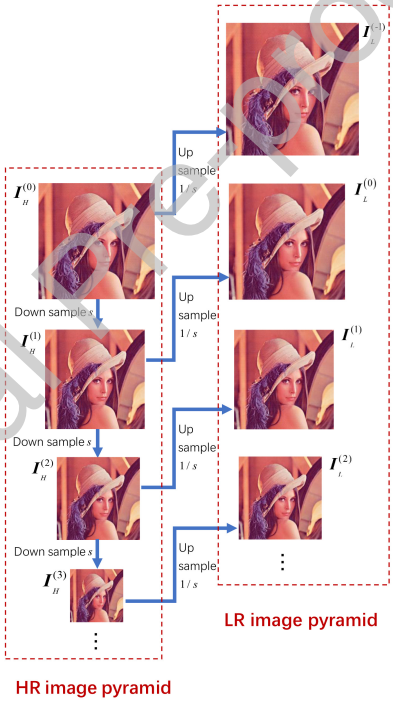

Fig. 1: Generation of the image pyramids.

As shown in Fig. 1, we generate an HR image pyramid $\boldsymbol{\Psi}_H = \left\{\boldsymbol{I}_H^{(\varphi)}\right\}_{\varphi=0}^{l\text{-}1}$ and its corresponding LR image pyramid $\boldsymbol{\Psi}_L = \left\{\boldsymbol{I}_L^{(\varphi)}\right\}_{\varphi=-1}^{l-2}$, as in [41, 50, 28], where $\boldsymbol{I}_H^{(\varphi)}$ and $\boldsymbol{I}_L^{(\varphi)}$ are the φth layer images of the HR and LR image

pyramids, respectively. The input LR image $\boldsymbol{I}_{L}$ is used as the first layer of the HR image pyramid, and $\boldsymbol{I}_{H}^{(0)} = \boldsymbol{I}_{L}$. Then, the other image layers in the HR image pyramid are generated by down-sampling $\boldsymbol{I}_{L}$. The down-sampling factor is s $(0 < s < 1)$. LR image pyramid $\boldsymbol{\Psi}_{L}$ is generated by up-sampling the image layers in HR image pyramid $\boldsymbol{\Psi}_{H}$. The up-sampling factor is $1/s$. We use $\boldsymbol{I}_{L}^{(-1)}$ as the LR testing image and the others as training images. The testing set and training set are produced by dividing the testing image and training images into patches.

2.2. Extract Features

The patch features determine the efficiency of the patch matching process and the quality of the matching patches. To reduce the time complexity of the patch matching process, the patch feature dimensions should be as short as possible in order to reduce the computational complexity. To improve the quality of the matching patches, the patch features should be able to distinguish similar LR patches so that precise HR patches can be found for reconstruction.

2.2.1. Variance feature

The variance feature[43] calculates the variance of each patch. Since it is only one dimension, it is short enough to reduce the time complexity and be able to distinguish texture and plane patches. However, it is unable to distinguish similar LR patches. Therefore, this feature is only used to perform coarse selection.

2.2.2. Principal component analysis (PCA) feature

Principal component analysis(PCA) is an effective dimensional reduction method. The convolutional principal component analysis(CPCA) method that was proposed in [43] has lower computational complexity and lower time costs than the traditional PCA feature extraction method. Therefore, we use the CPCA method to extract the PCA features of the LR image pyramid. We use the CPCA method to extract the PCA features of each layer in the LR pyramid and obtain the PCA feature pyramid $\hat{\boldsymbol{\Psi}}_{L} = \left\{\hat{\boldsymbol{I}}_{L}^{(\varphi)}\right\}_{\varphi=-1}^{l-2}$, where $\hat{\boldsymbol{I}}_{L}^{(\varphi)}=\{\hat{\boldsymbol{I}}_{L}^{p,(\varphi)}\}_{p=1}^{d}$ is the φth layer PCA feature pyramid. $\hat{\boldsymbol{I}}_{L}^{p,(\varphi)}$ is the pth feature in the φth layer image of the PCA feature pyramid. The extraction process is shown in Fig. 2. Suppose that $\boldsymbol{E} = [\boldsymbol{e}^{(1)}, ..., \boldsymbol{e}^{(\delta)}, ..., \boldsymbol{e}^{(d)}]$ is a PCA dictionary. $\boldsymbol{e}^{(\delta)}$ is the δth dictionary atom. d is the number of atoms. The

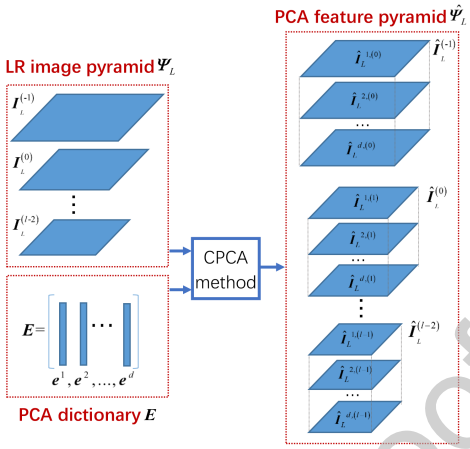

Fig. 2: Generation of the PCA pyramid.

main idea of CPCA is to convert every PCA dictionary atom into a filter. Then, we convolve the filter with every image in the LR pyramid $\boldsymbol{\Psi}_L$ to get the PCA pyramid $\hat{\boldsymbol{\Psi}}_L$.

2.3. Finding candidate patch

Finding a similar patch in an LR data set is exhaustive work. A traditional method, such as NE, finds matching patches using the whole data set or a small data region (as shown in Fig. 3).

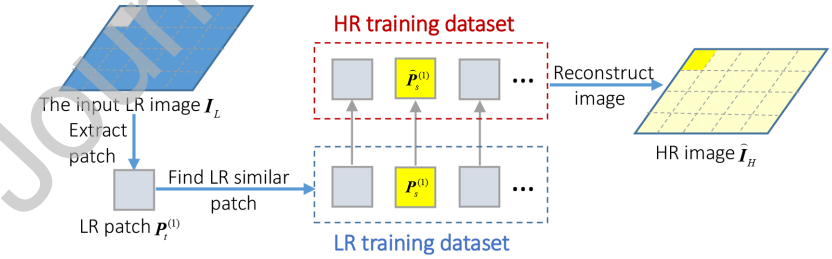

Fig. 3: Flowchart of NE.

NE finds similar patches in the LR data set and then uses the HR image patches that correspond to similar LR patches to reconstruct the HR image. CPCA[43] divides the input LR image $\boldsymbol{I}_L$ into patches. Suppose that $\boldsymbol{P}_t^{(1)}$ is one of the LR patches. We then find the similar patch $\boldsymbol{P}_s^{(1)}$ corresponding

to $\boldsymbol{P}_t^{(1)}$ in the LR data set and put its corresponding HR patch $\widehat{\boldsymbol{P}}_s^{(1)}$ at the same coordinate as $\boldsymbol{P}_t^{(1)}$. We perform the same operation on all of the test patches and reconstruct the HR image $\widehat{\boldsymbol{I}}_H$. The process of finding candidate patches is shown in Fig. 4. Suppose that the test patch $\boldsymbol{P}_t^{(k)}$ gets its matching patch $\boldsymbol{P}_s^{(k,N)}$ after the Nth iteration. Then we generate candidate patches $\tilde{\boldsymbol{P}}$ of matching patch $\boldsymbol{P}_s^{(k,N)}$ in the next iteration (the generation of candidate patches $\tilde{\boldsymbol{P}}$ will be described in Section 3.2), and compare candidate patches $\tilde{\boldsymbol{P}}$ with the existing matching patch $\boldsymbol{P}_s^{(k,N)}$ to select the best candidate patch $\boldsymbol{P}_s^{(k,N+1)}$.

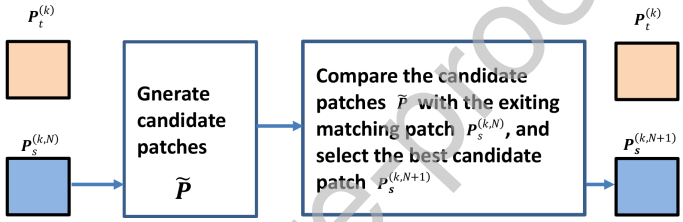

Fig. 4: Finding the best candidate patch.

3. Proposed method

The existing feature extraction methods and patch matching methods have two shortcomings. (1) The variance feature can only distinguish smooth and textured patches but cannot distinguish more textured features. When finding matching patches using NE, a short feature that can distinguish the texture of patches, which can decrease the computational complexity and increase the matching reliability,should be explored. (2) Using the traditional local /non-local similarity in the image pyramid to find matching patch candidates has not been fully explored. Therefore, we propose a new NE algorithm based on self-learning, as described in the following sections.

3.1. Extract LBP features

We extract the LBP features[51] to perform coarse selection. Compared with the traditional variance feature, the LBP feature is one dimensional and can sufficiently reflect the local textures of an image.

The LBP feature of a given coordinate (x_c, y_c) is as follows:

$$\boldsymbol{G}\left(x_c, y_c\right) = \sum_{q=0}^{7} 2^q \boldsymbol{S}\left(\mu_q - \mu_c\right) \quad q \neq c \tag{1}$$

where $\boldsymbol{G}\left(x_c, y_c\right)$ is the LBP feature of the center pixel whose position coordinate is (x_c, y_c). In the 3×3 neighborhood, μ_c is the center pixel of the neighborhood, μ_q is its surrounding pixel, and $\boldsymbol{S}\left(\bullet\right)$ is defined as follows:

$$\boldsymbol{S}\left(\mu_q - \mu_c\right) = \begin{cases} 1, & \mu_q > \mu_c \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

The original LBP operator takes the center pixel value as the threshold, and the surrounding pixels are respectively compared with the center pixel. If the surrounding pixel is larger than the center pixel value, the pixel is marked as 1; and otherwise, it is 0. The eight points are arranged clockwise or counter clockwise to generate 8-bit binary numbers and obtain a decimal number which is the LBP value of the center pixel. Fig. 5 shows an example of calculating the LBP feature. By putting the LBP feature of each pixel on its original pixel position, the LBP feature pyramid $\boldsymbol{\Psi}_{lbp} = \left\{ \breve{\boldsymbol{I}}_{L}^{(\varphi)} \right\}_{\varphi=-1}^{l-2}$ can be obtained, where $\breve{\boldsymbol{I}}_{L}^{(\varphi)}$ is the φth layer in $\boldsymbol{\Psi}_{lbp}$ that is generated by the LR image $\boldsymbol{I}_{L}^{(\varphi)}$.

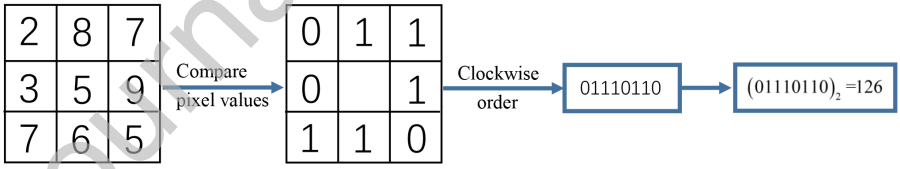

Fig. 5: The process of generating the LBP feature. Suppose that the center pixel value is 5 and its neighborhood pixels are 2,8,7,9,5,6,7 and 3. If the neighborhood pixel value is larger than 5, the flag is set to 1. Otherwise, the flag is set to 0. Then we can obtain a string of binary numbers 01110110. By converting 01110110 to a decimal number, we get 126. Therefore, 126 is the LBP feature of the center pixel.

3.2. Matching Patches

There are three steps to select the matching patch candidate: random oscillation, horizontal propagation and vertical propagation, the details will be described in the following sections.

3.2.1. Random oscillation

In the random oscillation step, the best candidate patch $\boldsymbol{P}_s^{(k,N+1)}$ is generated using a coarse to fine process. First, the candidate patches $\tilde{\boldsymbol{P}}$ are generated by randomly choosing their positions and layers in the feature pyramid $\hat{\boldsymbol{\Psi}}_L$. The best candidate patch $\boldsymbol{P}_s^{(k,N+1)}$ is chosen by comparing the LBP features (described in Section 3.1) of the candidate patches $\tilde{\boldsymbol{P}}$. Then, the best candidate patch $\boldsymbol{P}_s^{(k,N+1)}$ performs fine selection. Every image patch chooses its matching patch independently. Fig. 6 shows the framework of random oscillation.

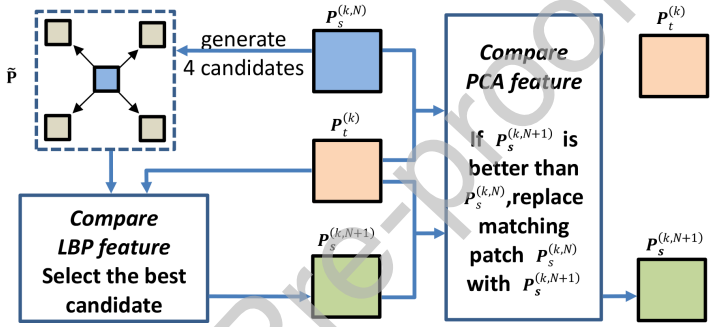

Fig. 6: Framework of random oscillation.

Suppose that the test patch $\boldsymbol{P}_t^{(k)}$ gets its matching patch $\boldsymbol{P}_s^{(k,N)}$ after the Nth iteration. We compare the LBP features of the candidate patches $\tilde{\boldsymbol{P}}$, and find the best candidate patch $\boldsymbol{P}_s^{(k,N+1)}$ in the next iteration. Then we compare the PCA features of the candidate patch $\boldsymbol{P}_s^{(k,N+1)}$and matching patch $\boldsymbol{P}_s^{(k,N)}$ with the test patch $\boldsymbol{P}_t^{(k)}$ independently. We use the city block distance to measure the similarity between patches. The similarity between $\boldsymbol{P}_s^{(k,N)}$ and $\boldsymbol{P}_t^{(k)}$ is $\ell_{(N)} = \left\| \boldsymbol{\alpha}_s^{(k,N)} - \boldsymbol{\alpha}_t^{(k)} \right\|_1$, where $\boldsymbol{\alpha}_s^{(k,N)}$ is the PCA feature of patch $\boldsymbol{P}_s^{(k,N)}$ and $\boldsymbol{\alpha}_t^{(k)}$ is the PCA feature of patch $\boldsymbol{P}_t^{(k)}$. The similarity between $\boldsymbol{P}_s^{(k,N+1)}$ and $\boldsymbol{P}_t^{(k)}$ is $\ell_{(N+1)} = \left\| \boldsymbol{\alpha}_s^{(k,N+1)} - \boldsymbol{\alpha}_t^{(k)} \right\|_1$, where $\boldsymbol{\alpha}_s^{(k,N+1)}$ is the PCA feature of patch $\boldsymbol{P}_s^{(k,N+1)}$. If $\ell_{(N+1)} < \ell_{(N)}$, the matching patch of $\boldsymbol{P}_t^{(k)}$ will be changed to $\boldsymbol{P}_s^{(k,N+1)}$.

3.2.2. Horizontal propagation

In the horizontal propagation step, the matching patch $\boldsymbol{P}_s^{(k,N+1)}$ is chosen according to the matching patch of its neighboring patches. The reason

for using horizontal propagation is the local similarity prior[41]. The local similarity prior of images tells us that an image patch is similar to its neighborhood patches. Therefore, the best matching patch information can be propagated to its neighborhood patches. Fig. 7 shows the framework of Horizontal propagation.

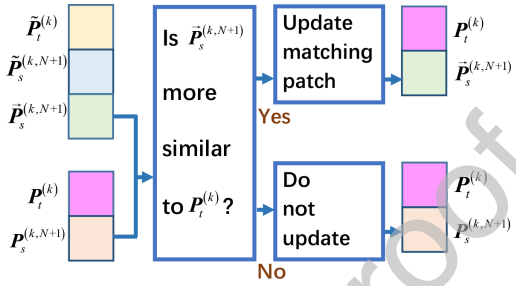

Fig. 7: Framework of horizontal propagation.

Suppose that $\boldsymbol{P}_s^{(k,N+1)}$ is the matching patch of test patch $\boldsymbol{P}_t^{(k)}$ in the $(N+1)$th iteration. $\boldsymbol{P}_t^{(k)}$ has an adjacent patch $\tilde{\boldsymbol{P}}_t^{(k)}$ in its neighborhood. $\tilde{\boldsymbol{P}}_t^{(k)}$ also gets its matching patch $\tilde{\boldsymbol{P}}_s^{(k,N+1)}$ in the $(N+1)$th iteration. $\vec{\boldsymbol{P}}_s^{(k,N+1)}$ is an adjacent patch of $\tilde{\boldsymbol{P}}_s^{(k,N+1)}$. We compare $\boldsymbol{P}_s^{(k,N+1)}$ and $\vec{\boldsymbol{P}}_s^{(k,N+1)}$ with $\boldsymbol{P}_t^{(k)}$. If $\vec{\boldsymbol{P}}_s^{(k,N+1)}$ is more similar to $\boldsymbol{P}_t^{(k)}$, the matching patch $\boldsymbol{P}_s^{(k,N+1)}$ is changed to $\vec{\boldsymbol{P}}_s^{(k,N+1)}$.

This means that the matching patch of the test patch is propagated to its neighborhood. This horizontal propagation only executes on the same layer of the image pyramid.

3.2.3. Vertical propagation

We find that the patches in the same position on different layers in the image pyramid are similar (shown in Fig. 8). We call it 'vertical similarity'.

According to 'vertical similarity', we propose 'vertical propagation' to find matching patch candidates (shown in Fig. 9).

Suppose $\boldsymbol{P}_s^{(k,N+1)}$ is the best matching patch of $\boldsymbol{P}_t^{(k)}$ in the $(N+1)$th iteration, and we record the coordinate of $\boldsymbol{P}_s^{(k,N+1)}$ as $(x_s^{(k,N+1)}, y_s^{(k,N+1)})$. We find the test patch $\boldsymbol{P}_t^{(\mu)}$ whose coordinate is also $(x_s^{(k,N+1)}, y_s^{(k,N+1)})$. Suppose that $\boldsymbol{P}_s^{(\mu,N+1)}$ is the matching patch of $\boldsymbol{P}_t^{(\mu)}$. Because of the vertical similarity, we can compare $\boldsymbol{P}_s^{(\mu,N+1)}$ with $\boldsymbol{P}_s^{(k,N+1)}$. We still use the PCA features

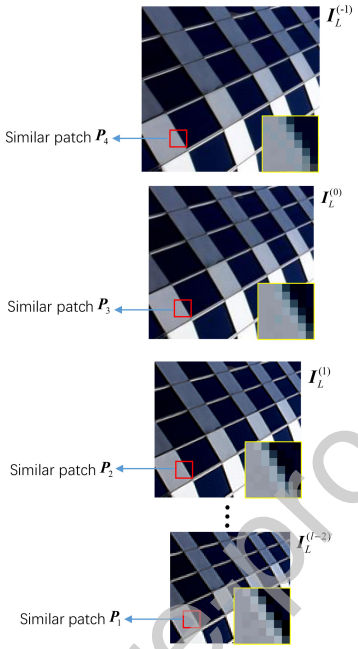

Fig. 8: Vertical similarity. The marked patches $\boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{P}_3$, and $\boldsymbol{P}_4$ are at the same position but on different layers in the LR pyramid. When the magnification factor between layers s is small, these patches are similar.

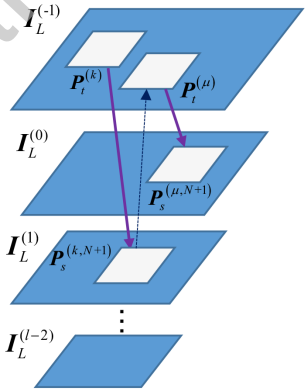

Fig. 9: Framework of vertical propagation.

to compare the similarity between $\boldsymbol{P}_s^{(\mu,N+1)}$ and $\boldsymbol{P}_t^{(k)}$ to determine whether to change the matching patch of $\boldsymbol{P}_t^{(k)}$. If $\boldsymbol{P}_s^{(\mu,N+1)}$ is more similar to $\boldsymbol{P}_t^{(k)}$

than to $\boldsymbol{P}_s^{(k,N+1)}$, we change the matching patch of $\boldsymbol{P}_t^{(k)}$ to $\boldsymbol{P}_s^{(\mu,N+1)}$. This propagation executes on the different layers. Thus, we propagate the information of the best matching patch using the similarity relationship between different scale patches in the vertical direction.

3.3. Algorithm summary

As shown in Table 2 and Fig.10, the proposed algorithm contains training and testing stages. The training stage includes: the input image $\boldsymbol{I}_L$, construction of the image pyramid and construction of the feature pyramid. The testing stage includes: preparation of the LR test image, preparation of the test feature map, random oscillation, horizontal propagation, vertical propagation, HR image reconstruction and IBP.

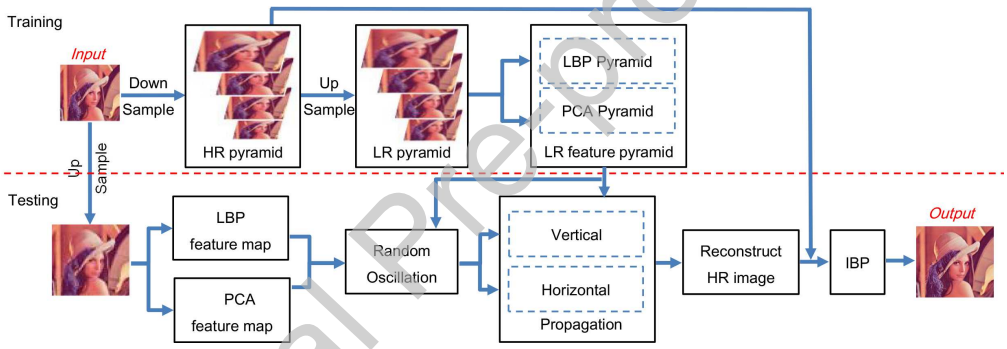

Fig. 10: Algorithm flowchart

4. Experimental results

4.1. Experimental setting

To compare the proposed algorithm with the state-of-the-art SR methods, we use the Set5, Set14 and Urban100 data sets, which are obtained from the code package[28]. Set5 and Set14 contain animals, humans, and scenery. Urban100 contains more buildings with repetitive structures. We use an AMD Ryzen 1600X CPU (3.6GHz) and 16GB of memory, and only the CNN methods are performed using a GTX1070Ti Graphics Processing Unit (GPU).

The experimental parameters are set as follows. The patch size $\sqrt{a} \times \sqrt{a}$ is set to 7×7, and the number of dictionary atoms d is set to 10. According

Table 2: Algorithm Steps

Algorithm: The Proposed Algorithm

1.Training:

(1)Input image $\boldsymbol{I}_{L}$.

(2)Construct the image pyramid: Construct HR pyramid $\boldsymbol{\Psi}_H$ and LR pyramid $\boldsymbol{\Psi}_L$ by down-sampling and up-sampling the input image $\boldsymbol{I}_{L}$ (as described in Section 2.1).

(3)Construct the feature pyramid: Construct the LBP feature pyramid $\boldsymbol{\Psi}_{lbp} = \left\{ \breve{\boldsymbol{I}}_{L}^{(\varphi)} \right\}_{\varphi=0}^{l-2}$ (as described in Section 3.1)and the PCA feature pyramid $\hat{\boldsymbol{\Psi}}_L = \left\{ \hat{\boldsymbol{I}}_{L}^{(\varphi)} \right\}_{\varphi=0}^{l-2}$ via the LR pyramid $\boldsymbol{\Psi}_L$ (as described in Section 2.2.2).

2.Testing:

(1)Prepare the LR test image: Up-sample the input image $\boldsymbol{I}_{L}$ and obtain the LR test image $\boldsymbol{I}_{L}^{(-1)}$.

(2)Prepare the test feature map: Calculate the LBP feature map $\breve{\boldsymbol{I}}_{L}^{(-1)}$ (as described in Section 3.1)and the PCA feature pyramid $\hat{\boldsymbol{I}}_{L}^{(-1)}$ (as described in Section 2.2.2).

(3)Random oscillation: Generate candidate patches $\tilde{\boldsymbol{P}}$ for each test patch using random oscillation. Select the best candidate patch $\boldsymbol{P}_s^{(k,N+1)}$ by comparing the LBP feature, and update the matching patch of the test patch $\boldsymbol{P}_t^{(k)}$ with $\boldsymbol{P}_s^{(k,N+1)}$. Repeat this step until the end of the iteration (as described in Section 3.2.1).

(4)Horizontal propagation: Update the matching patch of the neighborhood of $\boldsymbol{P}_t^{(k)}$ by using the horizontal similarity prior. Repeat this step until the end of the iteration. (as described in Section 3.2.2).

(5)Vertical propagate: Update the matching patch of $\boldsymbol{P}_t^{(k)}$ using the vertical similarity prior, repeat this step until the end of the iteration. (as described in Section 3.2.3).

(6)Reconstruct HR image $\widehat{\boldsymbol{I}_H}$: Find the HR training patches corresponding to selected LR matching patches and reconstruct the HR image $\widehat{\boldsymbol{I}_H}$.

(7)IBP: Enhance the image quality with iterative back projection (IBP)[52].

to [43], if the SR magnification factor is an integer multiple of 2, then the sampling factor s is set to 0.7937. Otherwise, s is set to 0.8027. We did experiments for 2x, 3x and 4x magnification. For 2x and 4x magnification, we set 3 layers for every 2x magnification. $\sqrt[3]{1/2} = 0.7937$. So the sampling factor s is set to 0.7937 for 2x and 4x magnification. For 3x magnification, we set 5 layers for every 3x magnification. $\sqrt[5]{1/3} = 0.8027$ .So the sampling factor s is set to 0.8027. If the size of the original image is $m \times n$, the initial oscillating horizontal radius $r = \frac{max(m,n)}{4}$. The number of vertical propagation iterations is set to 2. (more data is provided in the supplementary materials.)

4.2. Comparison

We compare the proposed algorithm with common self-learning methods and some external learning methods to verify its performance. These algorithms include the bi-cubic interpolation algorithm [53], Glasner's algorithm[50], the super resolution convolutional neural network (SR-CNN) algorithm[54], the clustering and collaborative representation (CCR) algorithm[55], the very deep super resolution convolutional neural network (VDSR) algorithm[56], the Laplacian pyramid super-resolution network (LAPSRN)[57] and Jia-Bin Huang's algorithm[28]. We use the peak signal-to-noise ratio (PSNR) and the structural similarity index (SSIM)[58] as the quality assessment standards.

As shown in Table 3, the proposed algorithm gets the highest PSNRs in the self-learning group for the Urban100 data set. The visual comparison of the experimental results is shown in Fig.11-Fig.14. To help in visual perception of errors, we also compared error maps. The error maps are pixel value errors between output images and their corresponding original HR images. These results show that our algorithm is more effective for images with highly repetitive structures. If an image has a highly repetitive structure, the image contains more similar patches. It gets high probability to find matching patches in random oscillation step. The proposed algorithm is inferior to the VDSR and LAPSRN algorithms, but it does not need an external data set or a GPU.

We also compared our algorithm with some methods based on deep learning. They are TTSR[19], DRN[20], MZSR[59], DAN[16] and SPSR[18]. As shown in Table 4 and Fig.15-Fig.16, though the PSNR and SSIM values are inferior to methods based on deep learning, our algorithm does not occupy storage space to store a trained model. This is useful in some application areas.

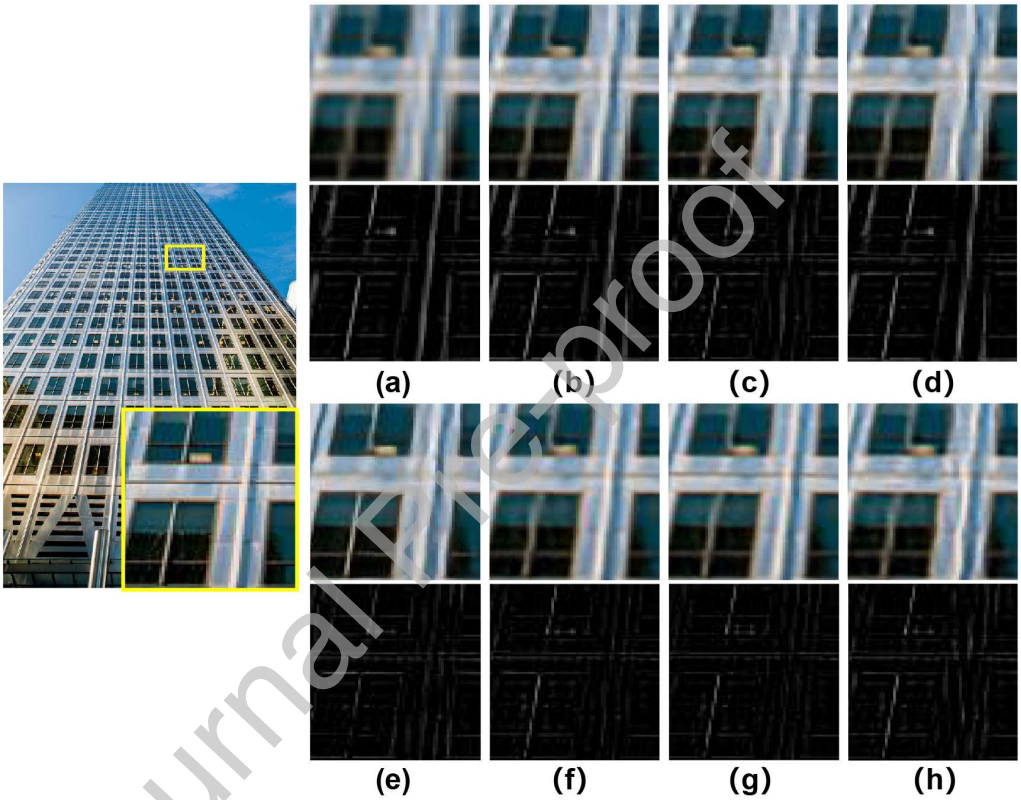

Fig. 11: The 30th Urban100 image (3x) of SR algorithms, the top images are outputs and the bottom images are error maps. (a) Bi-cubic. PSNR=21.5967, SSIM=0.6975 (b) A+. PSNR=23.0451, SSIM=0.7874 (c) SR-CNN. PSNR=22.9982, SSIM=0.7804 (d) CCR. PSNR=22.9450, SSIM=0.7833 (e) VDSR. PSNR=23.9516, SSIM=0.8294 (f) Glasner. PSNR=23.5849, SSIM=0.8100 (g) Jia-Bin Huang. PSNR=24.0531, SSIM=0.8342 (h) Proposed. PSNR=24.3708, SSIM=0.8489.

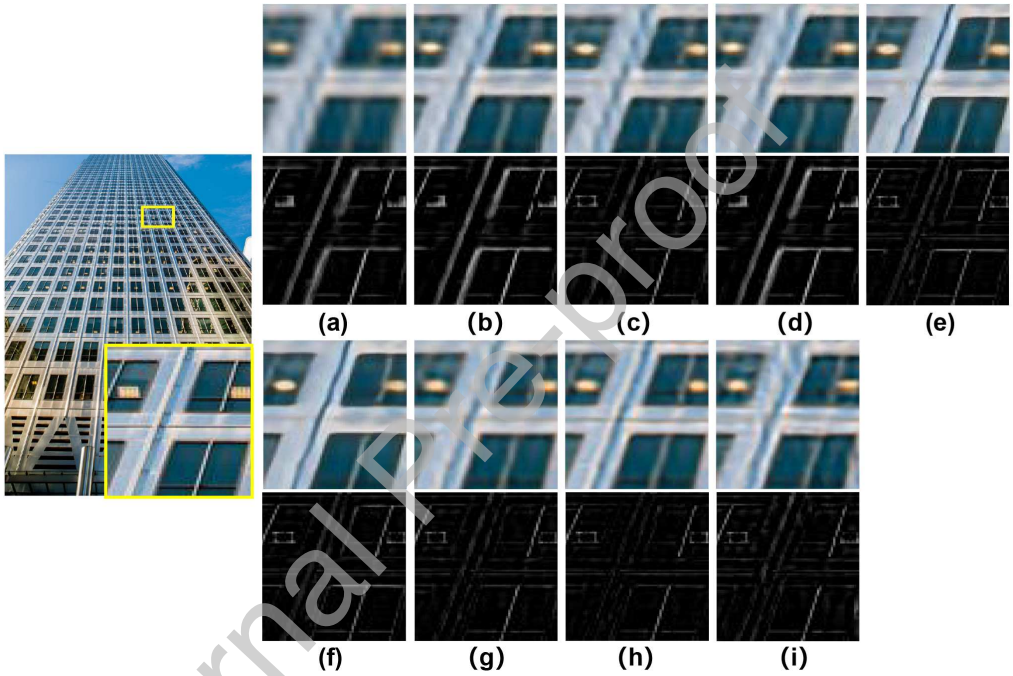

Fig. 12: The 30th Urban100 image (4x) of SR algorithms, the top images are outputs and the bottom images are error maps. (a) Bi-cubic. PSNR=21.7654, SSIM=0.6931 (b) A+. PSNR=21.4204, SSIM=0.7874 (c) SR-CNN. PSNR=21.3284, SSIM=0.6751 (d) CCR. PSNR=21.3776, SSIM=0.6902 (e) VDSR. PSNR=22.1349, SSIM=0.7363 (f) LAPSRN. PSNR=23.2129, SSIM=0.8219 (g) Glasner. PSNR=22.0360, SSIM=0.7411 (h) Jia-Bin Huang. PSNR=22.2983, SSIM=0.7515 (i) Proposed. PSNR=22.5321, SSIM=0.7668.

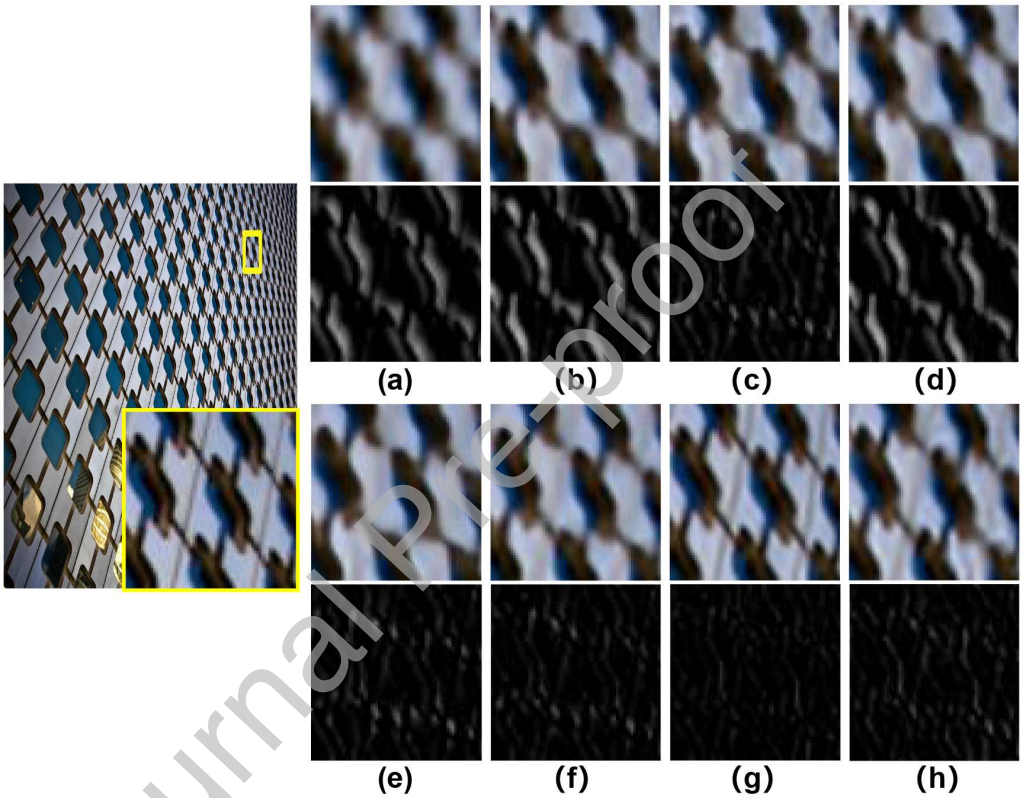

Fig. 13: The 41th Urban100 image (3x) of SR algorithms, the top images are outputs and the bottom images are error maps. (a) Bi-cubic. PSNR=20.7615, SSIM=0.7484 (b) A+. PSNR=23.2026, SSIM=0.8486 (c) SR-CNN. PSNR=22.8281, SSIM=0.8242 (d) CCR. PSNR=23.1618, SSIM=0.8461 (e) VDSR. PSNR=24.7949, SSIM=0.8942 (f) Glasner. PSNR=24.2962, SSIM=0.8766 (g) Jia-Bin Huang. PSNR=26.0455, SSIM=0.9090 (h) Proposed. PSNR=27.4664, SSIM=0.9263.

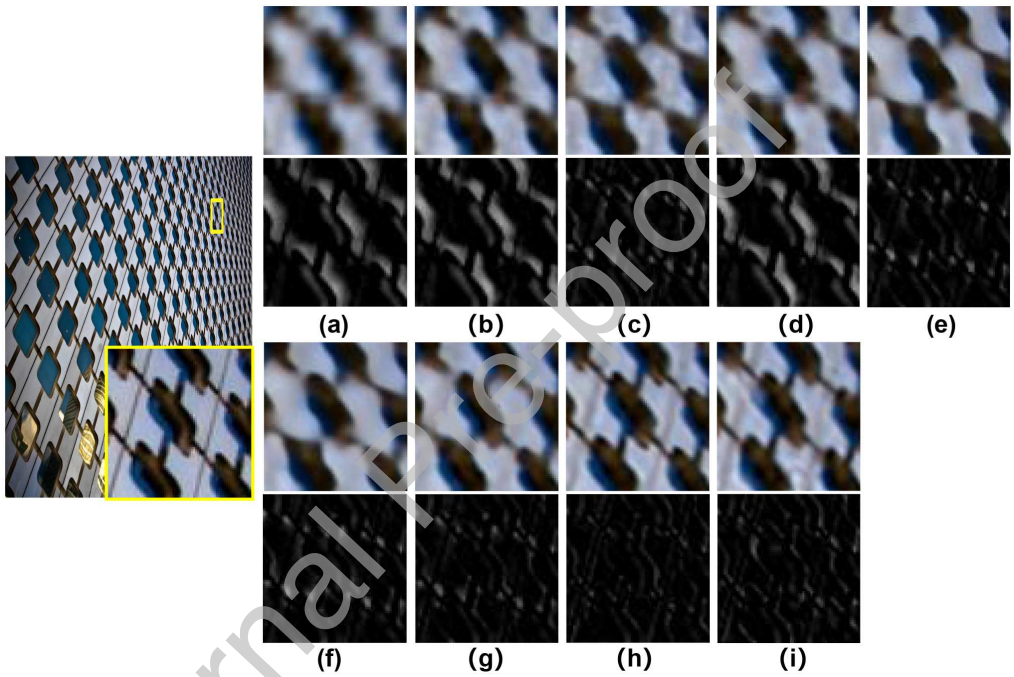

Fig. 14: The 41th Urban100 image (4x) of SR algorithms, the top images are outputs and the bottom images are error maps. (a) Bi-cubic. PSNR=21.3655, SSIM=0.7589 (b) A+. PSNR=21.6471, SSIM=0.7576 (c) SR-CNN. PSNR=21.2980, SSIM=0.7266 (d) CCR. PSNR=21.5524, SSIM=0.7518 (e) VDSR. PSNR=23.0145, SSIM=0.8157 (f) LAPSRN. PSNR=23.2129, SSIM=0.8219 (g) Glasner. PSNR=22.7715, SSIM=0.8023 (h) Jia-Bin Huang. PSNR=23.2262, SSIM=0.8194 (i) Proposed. PSNR=25.4055, SSIM=0.8656.

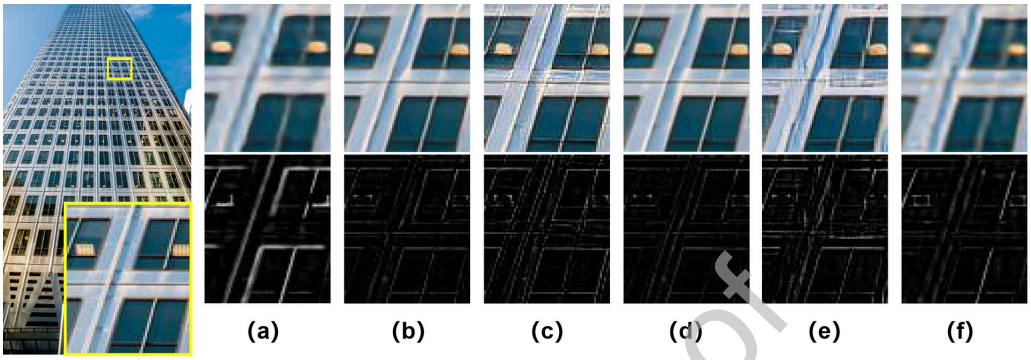

Fig. 15: The 30th Urban100 image (4x) of SR algorithms, the top images are outputs and the bottom images are error maps. (a) MZSR. PSNR=18.8636, SSIM=0.5212 (b) DAN. PSNR=24.8522, SSIM=0.8245 (c) SPSR. PSNR=22.9766, SSIM=0.7726 (d) DRN. PSNR=25.5181, SSIM=0.8497 (e) TTSR. PSNR=22.3880, SSIM=0.7331 (f) Proposed. PSNR=22.8888, SSIM=0.7090.

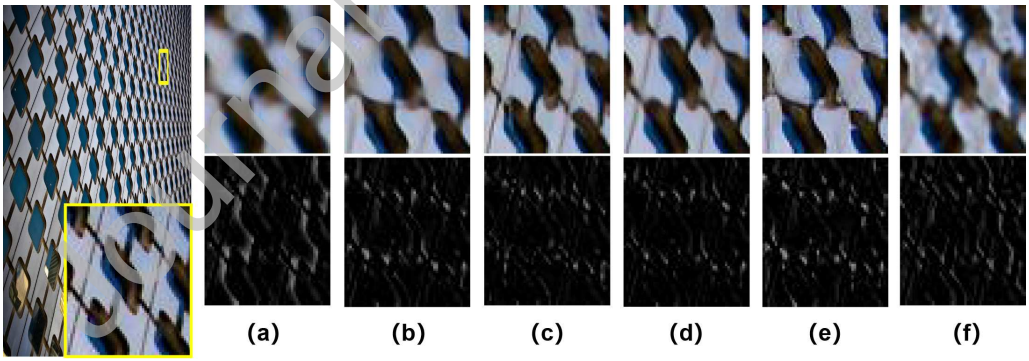

Fig. 16: The 41th Urban100 image (4x) of SR algorithms, the top images are outputs and the bottom images are error maps. (a) MZSR. PSNR=18.2657, SSIM=0.5953 (b) DAN. PSNR=26.1229, SSIM=0.8812 (c) SPSR. PSNR=24.7800, SSIM=0.7464 (d) DRN. PSNR=27.4092, SSIM=0.9048 (e) TTSR. PSNR=24.6086, SSIM=0.7465 (f) Proposed. PSNR=24.2978, SSIM=0.7011.

Table 3: Performance comparison of the different algorithms. We have three rows in each table box. The first row is the PSNR value (dB), the second row is the SSIM value, and the third row is the time cost (s). The best performance in the external-learning methods is marked in bold. The best performance in the self-learning methods is underlined.

Factor	Database	Bi-cubic	External-learning					Self-learning		
			A+	SR-CNN	CCR	VDSR	LAPSRN	Glasner	Jia-Bin Huang	Proposed
2x	Urban100	25.1725	27.4550	27.3356	27.3675	**28.9301**	28.6411	27.7229	27.9546	28.6382
		0.8265	0.8879	0.8836	0.8849	**0.9094**	0.9061	0.8892	0.9069	0.9072
		0.0021	1.0168	6.4121	0.9131	0.1179	**0.0861**	120.3207	117.4202	37.3965
	Set5	32.3219	35.2261	35.0220	35.1533	36.1814	**36.1836**	35.0288	35.1486	35.4872
		0.9219	0.9490	0.9458	0.9468	0.9530	**0.9535**	0.9469	0.9483	0.9489
		0.0011	0.5610	2.9076	0.5517	**0.0930**	0.2508	66.7788	58.8712	10.6983
	Set14	29.0118	31.0890	31.0211	31.0108	**31.9053**	31.8477	30.9720	31.1205	31.4698
		0.8572	0.8968	0.8947	0.8934	0.9038	**0.9040**	0.8946	0.8969	0.8974
		0.0023	1.1839	7.3676	1.0811	**0.1399**	0.2176	156.4127	138.7021	34.6663
3x	Urban100	23.1197	24.8915	24.6995	24.7852	**25.9906**	-	25.2333	25.4609	26.2676
		0.7170	0.7923	0.7812	0.7871	**0.8240**	-	0.8105	0.8210	0.8346
		0.0025	1.3518	14.7927	1.2013	**0.2619**	-	252.4793	205.0911	90.3269
	Set5	29.0900	31.2805	31.1431	31.2784	**32.3989**	-	31.3644	31.3702	31.7796
		0.8555	0.9002	0.8924	0.8971	**0.9122**	-	0.8986	0.8992	0.9013
		0.0012	0.3637	2.8820	0.3462	**0.0878**	-	56.0918	42.3480	10.7681
	Set14	26.3070	27.9426	27.8217	27.8424	**28.5935**	-	27.9054	27.9852	28.3394
		0.7552	0.8043	0.7991	0.8008	**0.8166**	-	0.8029	0.8065	0.8092
		0.0016	0.7068	7.3116	0.6624	**0.1361**	-	129.0513	96.9856	35.1767
4x	Urban100	21.8044	22.9978	22.8045	22.9191	23.8359	**23.8630**	23.2952	23.3622	24.1285
		0.6346	0.6999	0.6847	0.6949	0.7353	**0.7387**	0.7104	0.7366	0.7632
		0.0031	1.8059	28.3236	1.7575	0.3062	**0.2915**	398.6492	335.3643	170.3594
	Set5	27.1300	29.0920	28.8023	28.9632	30.1106	**30.2831**	29.0896	28.9937	29.3514
		0.7935	0.8490	0.8373	0.8447	0.8714	**0.8740**	0.8493	0.8480	0.8536
		0.0015	0.2594	2.9150	0.2574	**0.0854**	0.4399	44.4942	37.6559	9.9337
	Set14	24.7647	26.1053	25.9959	25.0506	26.8319	**26.8865**	26.1809	26.2154	26.5746
		0.6781	0.7293	0.7198	0.7258	0.7476	**0.7504**	0.7302	0.7337	0.7348
		0.0019	0.5388	7.3501	0.5342	**0.1429**	0.2263	101.1306	84.6965	33.9812

Table 4: Performance comparison of different algorithms. In each table box, the first line is PSNR value (dB). The second line is SSIM. The third line is the space size of the model (MB).

Methods	MZSR	DAN	SPSR	DRN	TTSR	Proposed
PSNR	20.3687	26.2150	24.7800	**26.8133**	24.6086	24.2978
SSIM	0.5878	0.7924	0.7464	**0.8072**	0.7465	0.7011
Model	2.72	17.5	19.3	99.5	25.7	**0**

4.3. Parameter selection

4.3.1. Effectiveness of features

Since the above comparisons show that the proposed algorithm gets good results on the Urban100 data set, we perform feature selection experiments on the top 10 images of Urban100 (shown in Fig. 17). The quantity comparison

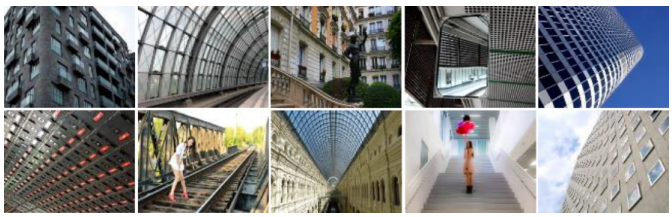

Fig. 17: Top 10 images in Urban100 data set. From left to right and top to bottom are Image 1-Image 10.

is shown in Table 5 and the visual quality is shown in Fig. 18.

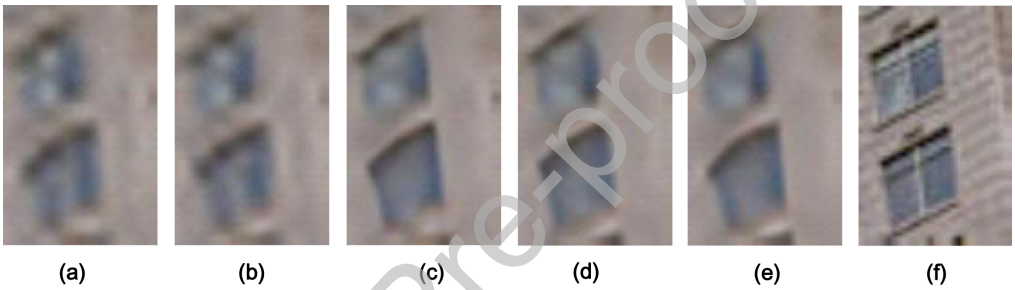

Fig. 18: Result images (4x) using different features in random oscillation phase. (a) variance feature. PSNR=19.6036, SSIM=0.6149 (b) LBP feature. PSNR=19.6500, SSIM=0.6215 (c) PCA feature. PSNR=19.8757, SSIM=0.6482 (d) variance + PCA. PSNR=19.8776, SSIM=0.6483 (e) LBP + PCA. PSNR=19.9077, SSIM=0.6496 (f) The original image.

Table 5: Performance comparison of different features. In each table box, the first line is PSNR value (dB). The second line is time cost (s).

Factor	Coarse (var)	Coarse (LBP)	Detail (PCA)	Coarse(var) +Detail	Coarse(LBP) +Detail
2x	26.9343 34.2158	27.3837 32.9029	28.7835 **28.0116**	28.8078 49.9064	**28.9273** 38.9885
3x	24.5004 66.3295	24.8614 63.3067	25.1734 **52.0507**	26.1829 74.5078	**26.2060** 73.6107
4x	22.8985 112.4753	23.0377 108.3294	23.9930 **90.4249**	24.1821 130.2263	**24.2067** 126.1847

We use coarse-to-detailed matching to select the matching patches. The

LBP or variance feature is used for coarse matching, and the PCA feature is used for detailed matching. For a fair comparison, we only change the features in the two-step random oscillation and only choose horizontal propagation in the propagation stage. If we only use the coarse matching method, the PSNR when using LBP is higher than when using the variance, and the time costs of extracting the LBP feature are lower than those when extracting the variance feature. When combining coarse and detailed selection, using the LBP feature also obtains better results.

4.3.2. Effects of different patch matching strategies

To compare the different patch matching strategies, we perform experiments on horizontal propagation, vertical propagation, and random oscillation (LBP+PCA), respectively, and compare combined versions of the above three methods. As shown in Table 6, the best result and visual effect are obtained for Method 8.

Table 6: Performance comparison of different patch matching strategies. It includes the horizontal propagation method (Method 1), the vertical propagation method (Method 2), the two-step oscillation (LBP+PCA) method (Method 3), the oscillation+horizontal propagation method (Method 4), the oscillation+vertical propagation method (Method 5), the oscillation+vertical propagation+horizontal propagation method (Method 6), the oscillation+horizontal propagation+vertical propagation method (Method 7) and the oscillation+vertical propagation+horizontal propagation+vertical propagation+horizontal propagation method (Method 8). In each table box, the first line is the PSNR value (dB). The second line is the SSIM value and the third line is the time cost (s).

Factor	Method1	Method2	Method3	Method4	Method5	Method6	Method7	Method8
2x	26.7023	26.7277	27.7277	28.9273	27.7706	28.9299	28.8864	**29.0237**
	0.8755	0.8761	0.8940	0.9138	0.8946	0.9138	0.9132	**0.9148**
	6.9090	8.2228	33.4066	38.9885	38.4238	41.3982	40.9696	62.7975
3x	24.3336	24.3894	25.2795	26.2060	25.3012	26.3240	26.2418	**26.3762**
	0.7700	0.7723	0.8009	0.8336	0.8013	0.8361	0.8346	**0.8369**
	17.9906	20.7225	65.5420	73.6107	72.3958	78.5283	80.0024	115.7195
4x	22.7379	22.7805	23.4982	24.2067	23.5039	24.2993	24.2516	**24.3385**
	0.6807	0.6831	0.7133	0.7532	0.7126	0.7562	0.7545	**0.7571**
	37.5597	37.6505	111.1490	126.1847	118.9431	130.3677	131.6034	194.7216

For the 3× and 4× magnifications, Methods 6 and 7 obtain better results than Method 4 (as shown in Fig.19).

This means that the vertical similarity prior can improve the quality of patch matching.

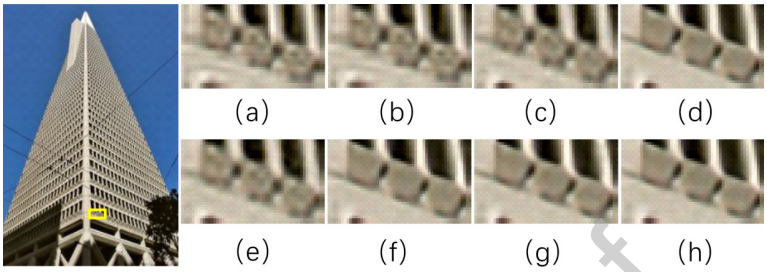

Fig. 19: The visual comparison of different patch matching strategies (3x). (a) Method 1. PSNR=19.0319, SSIM=0.7698 (b) Method 2. PSNR=19.0675, SSIM=0.7716 (c) Method 3. PSNR=19.5939, SSIM=0.7995 (d) Method 4. PSNR=21.5742, SSIM=0.8694 (e) Method 5. PSNR=19.5896, SSIM=0.7993 (f) Method 6. PSNR=21.6844, SSIM=0.8738 (g) Method 7. PSNR=21.6052, SSIM=0.8711 (h) Method 8. PSNR=21.7959, SSIM=0.8766.

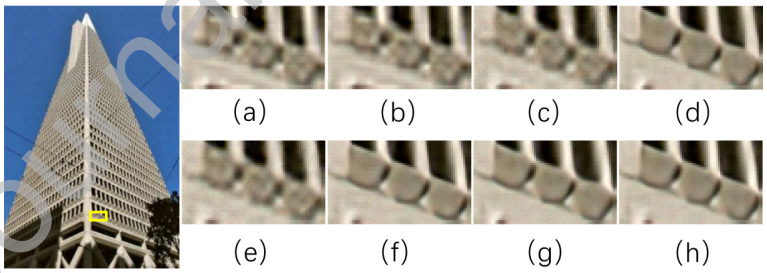

Fig. 20: The visual comparison of different patch matching strategies (4x). (a) Method 1. PSNR=17.2838, SSIM=0.6933 (b) Method 2. PSNR=17.3119, SSIM=0.6953 (c) Method 3. PSNR=17.7308, SSIM=0.7268 (d) Method 4. PSNR=19.2973, SSIM=0.8069 (e) Method 5. PSNR=17.7233, SSIM=0.7269 (f) Method 6. PSNR=19.3104, SSIM=0.8099 (g) Method 7. PSNR=19.2959, SSIM=0.8078 (h) Method 8. PSNR=19.3818, SSIM=0.8125.

Table 7: Performance comparison of the proposed method among different images in Urban 100. It includes the oscillation+ horizontal propagation method (Method 4) and the oscillation+ vertical propagation+ horizontal propagation method (Method 6). In each table box, the first line is the PSNR value (dB). The second line is the SSIM value and the third line is the time cost (s).

Factor	Propagation mode	Image1	Image2	Image3	Image4	Image5	Image6	Image7	Image8	Image9	Image10
2x	Method 4	30.4351	29.1632	26.4595	24.7316	31.4475	25.2193	29.6081	24.4539	34.8339	31.7469
		0.9164	0.9259	0.8565	0.9182	0.9805	0.8453	0.9338	0.8522	**0.9483**	0.9505
		51.5490	59.1565	54.8520	52.9350	60.7168	80.5159	53.2670	45.1996	64.7677	62.7906
	Method 6	**30.6380**	**29.4344**	**26.4951**	**24.7841**	**31.8682**	**25.3096**	**29.6705**	**24.6008**	**34.9452**	**32.1577**
		0.9178	**0.9301**	**0.8573**	**0.9185**	**0.9816**	**0.8489**	**0.9342**	**0.8552**	0.9468	**0.9527**
		49.9419	63.7807	57.8619	55.7656	65.7299	85.3148	57.5849	48.9797	69.2176	67.4813
3x	Method 4	**27.6721**	26.5768	23.7068	22.6543	29.1985	22.6108	27.7457	21.9107	**32.3335**	28.4328
		0.8238	0.8573	0.7289	0.8572	0.9635	0.7035	0.8693	0.7381	**0.9095**	0.9083
		86.4555	112.1590	98.6444	99.0053	115.7801	150.7651	98.4477	85.0666	126.1301	118.4818
	Method 6	27.5841	**26.7480**	**23.8278**	**22.7273**	**29.4315**	**22.6466**	**27.7766**	**21.9375**	32.3129	**28.4473**
		0.8212	**0.8629**	**0.7292**	**0.8599**	**0.9649**	**0.7045**	**0.8694**	**0.7389**	0.9093	**0.9084**
		93.0226	117.9353	105.4832	104.1670	122.5401	160.7468	104.8636	86.6665	126.3575	124.3076
4x	Method 4	25.2516	24.5861	21.4651	21.5006	27.1558	20.3515	26.2932	19.6858	**30.8761**	24.7697
		0.7108	0.7786	0.6011	0.7941	0.9438	0.5720	0.8050	0.6103	**0.8868**	0.8486
		150.7869	185.8002	165.5757	160.9437	190.7747	249.5491	165.5574	137.8230	210.1428	197.0080
	Method 6	**25.3391**	**24.7396**	**21.5430**	**21.5760**	**27.5554**	**20.4512**	**26.4615**	**19.8127**	30.6424	**25.0115**
		0.7106	**0.7857**	**0.6049**	**0.7969**	**0.9445**	**0.5805**	**0.8058**	**0.6205**	0.8784	**0.8529**
		156.9945	194.1552	173.4340	174.1813	197.7459	266.1863	168.7393	144.6607	214.9904	206.0915

To verify the effect of vertical propagation, we compare the top 10 images of the Urban100 data set using Method 4 and Method 6 in Table 7.

This table shows that vertical propagation is helpful for improving the quality of most of the images. However, when repetitiveness is not obvious, vertical propagation may fail (such as with image 9).

4.3.3. Effects of patch size and dictionary atom numbers

To choose a proper patch size and number of dictionary atoms, we did two experiments on Urban100 with 2x, 3x and 4x magnification. The results are shown in Table 8 and Table 9. The first experiment set patch size to be 7x7 and changed the number of the atoms to be 8, 10 and 12. As shown, when the number of the atoms is 10, the PSNR and SSIM values are the best. So, in the second experiment, we fixed the atom number to be 10 and changed the patch size to be 5x5, 7x7 and 9x9. When the patch size is 7x7, the PSNR and SSIM values are the best.

Table 8: Performance comparison of different dictionary atom numbers. We set the patch size $\sqrt{a} \times \sqrt{a}$ to be 7×7 and change the number of dictionary atoms d. In each table box, the first line is the average PSNR value (dB), the second line is the average SSIM value and the third line is the average time cost (s). The best performance is marked in bold.

Factor	d=8	d=10	d=12
2x	28.5982	**28.6382**	28.6141
	0.8865	**0.8873**	0.8864
	34.4263	37.3965	37.1179
3x	26.2241	**26.2676**	24.9629
	0.7940	**0.7951**	0.7816
	83.5233	90.3269	92.5946
4x	24.1208	**24.1285**	22.8419
	0.6948	**0.6964**	0.6752
	158.1511	170.3594	165.0279

Table 9: Performance comparison of different patch sizes. We set the number of dictionary atoms d to be 10 and change the patch size $\sqrt{a} \times \sqrt{a}$. In each table box, the first line is the average PSNR value (dB), the second line is the average SSIM value and the third line is the average time cost (s). The best performance is marked in bold.

Factor	5×5	7×7	9×9
2x	28.3523	**28.6382**	28.6344
	0.8744	**0.8873**	0.8882
	33.7602	37.3965	38.8277
3x	25.8645	**26.2676**	26.1138
	0.7694	**0.7951**	0.7902
	82.0753	90.3269	95.6407
4x	23.5169	**24.1285**	24.0522
	0.6508	**0.6964**	0.6913
	154.5763	170.3594	193.0403

5. Conclusion

This paper proposes a self-learning SR algorithm based on NE. We have shown that a good feature extraction method is helpful to improve the SR result. We use the LBP feature to select the best patch candidate for performing coarse selection, which eliminates the time complexity and improves the SR results; and a vertical similarity prior is found in the image pyramid. We propose a random oscillation+horizontal propagation+vertical propagation strategy to obtain better matching patches. Compared with traditional self-learning SR methods, the proposed algorithm obtains better results without using high-end equipment.

However, the proposed self-learning SR algorithm still cannot meet the needs of many real applications, with respect to both its effect and efficiency. In the future, we should explore self-learning algorithms that can obtain better results with relatively low time costs using low-end equipment.

Acknowledgment

This work was supported by the National Natural Science Foundation of China under Grant 62071380, 61601362, 41874173, 61571361, and 61671377, China Scholarship Council, Natural Science Basic Research Plan in Shannxi

Province of China 2019JQ-865, New Star Team of Xi'an University of Posts and Telecommunications xyt2016-01, Innovation Foundation for Postgraduate CXJJLZ202001.

References

[1] Z. Wang, J. Chen, S. C. H. Hoi, Deep learning for image super-resolution: A survey, IEEE Transactions on Pattern Analysis and Machine Intelligence In press (2021) 1–22.

[2] M. E. Helou, R. Zhou, S. Susstrunk, Stochastic frequency masking to improve super-resolution and denoising networks, in: European Computer Vision Association, Springer, Online, 2020, pp. 1–18.

[3] A. Lugmayr, M. Danelljan, L. Van Gool, R. Timofte, Srflow: Learning the super-resolution space with normalizing flow, in: European Conference on Computer Vision, Springer, Online, 2020, pp. 1–17.

[4] Y. Zhang, Z. Zhang, S. DiVerdi, J. E. Zhaowen Wang, Yun, Texture hallucination for large-factor painting super-resolution, in: European Conference on Computer Vision, Springer, Online, 2020, pp. 1–16.

[5] C. Tian, Y. Xu, W. Zuo, B. Zhang, L. Fei, C.-W. Lin, Coarse-to-fine cnn for image super-resolution, IEEE Transactions on Multimedia In press (2021) 1–14.

[6] B. Niu, W. Wen, W. Ren, X. Zhang, L. Yang, S. Wang, K. Zhang, X. Cao, H. Shen, Single image super-resolution via a holistic attention network, in: European Conference on Computer Vision, Springer, Online, 2020, pp. 1–16.

[7] K. Zhang, Z. Wang, J. Li, X. Gao, Z. Xiong, Learning recurrent residual regressors for single image super-resolution, Signal Processing 154 (2019) 324–337.

[8] K. Zhang, W. Zuo, L. Zhang, Learning a single convolutional super-resolution network for multiple degradations, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, Salt Lake City, Utah, USA.

[9] Z. Wang, B. Chen, H. Zhang, H. Liu, Variational probabilistic generative framework for single image super-resolution, Signal Processing 156 (2019) 92–105.

[10] Y. Cao, Z. He, Z. Ye, X. Li, Y. Cao, J. Yang, Fast and accurate single image super-resolution via an energy-aware improved deep residual network, Signal Processing 162 (2019) 115–125.

[11] W. Han, S. Chang, D. Liu, M. Yu, M. Witbrock, T. S. Huang, Image super-resolution via dual-state recurrent networks, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, Salt Lake City, Utah, USA, 2018, pp. 1654–1663.

[12] S. Anwar, N. Barnes, Densely residual laplacian super-resolution, IEEE Transactions on Pattern Analysis and Machine Intelligence In press (2021) 1–12.

[13] D. Zhang, J. Shao, Z. Liang, L. Gao, H. T. Shen, Large factor image super-resolution with cascaded convolutional neural networks, IEEE Transactions on Multimedia In press (2021) 1–14.

[14] J. Li, F. Fang, J. Li, K. Mei, G. Zhang, Mdcn: Multi-scale dense cross network for image super-resolution, IEEE Transactions on Circuits and Systems for Video Technology In press (2021) 1–15.

[15] Z. Zhang, X. Wang, C. Jung, Dcsr: Dilated convolutions for single image super-resolution, IEEE Transactions on Image Processing 28 (4) (2019) 1625–1635.

[16] Z. Luo, Y. Huang, S. Li, L. Wang, T. Tan, Unfolding the alternating optimization for blind super resolution, in: Conference and Workshop on Neural Information Processing Systems, Online, 2020, pp. 1–12.

[17] M. Zhang, Q. Ling, Supervised pixel-wise gan for face super-resolution, IEEE Transactions on Multimedia In press (2021) 1–13.

[18] C. Ma, Y. Rao, Y. Cheng, C. Chen, J. Lu, J. Zhou, Structure-preserving super resolution with gradient guidance, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, Online, 2020, pp. 7769–7778.

[19] F. Yang, H. Yang, J. Fu, H. Lu, B. Guo, Learning texture transformer network for image super-resolution, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, Online, 2020, pp. 5791–5800.

[20] Y. Guo, J. Chen, J. Wang, Q. Chen, J. Cao, Z. Deng, Y. Xu, M. Tan, Closed-loop matters: Dual regression networks for single image super-resolution, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, Online, 2020, pp. 5407–5416.

[21] Y. Xu, Z. Wu, J. Chanussot, Z. Wei, Nonlocal patch tensor sparse representation for hyperspectral image super-resolution, IEEE Transactions on Image Processing 28 (6) (2019) 3034–3047.

[22] Nikhil, A. Mhala, Pais, Contrast enhancement of progressive visual secret sharing (pvss) scheme for gray-scale and color images using super-resolution, Signal Processing 162 (2019) 253–267.

[23] W. Dong, L. Zhang, G. Shi, X. Li, Nonlocally centralized sparse representation for image restoration, IEEE Transactions on Image Processing, 22 (4) (2013) 1620–1630.

[24] B. Hou, K. Zhou, L. Jiao, Adaptive super-resolution for remote sensing images based on sparse representation with global joint dictionary model, IEEE Transactions on Geoscience and Remote Sensing 56 (4) (2018) 2312–2327.

[25] Z. Xu, Q. Ma, F. Yuan, Single color image super-resolution using sparse representation and color constraint, Journal of systems engineering and electronics 31 (2) (2020) 266–271.

[26] J. Hu, X. Wu, J. Zhou, Noise robust single image super-resolution using a multiscale image pyramid, Signal Processing 156 (2019) 92–105.

[27] J. Huang, P. Dragotti, Learning deep analysis dictionaries for image super-resolution, IEEE Transactions on Signal Processing 68 (2020) 6633–6648.

[28] J. Huang, A. Singh, N. Ahuja, Single image super-resolution from transformed self-exemplars, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, Boston, Massachusetts, USA, 2015, pp. 5197–5206.

[29] M. Yang, Y. F. Wang, A self-learning approach to single image super-resolution, IEEE Transactions on Multimedia 15 (3) (2013) 498–508.

[30] H. Wang, J. Li, Z. Dong, Single image super-resolution via self-similarity and low-rank matrix recovery, Multimedia Tools and Applications 77 (12) (2018) 15181–15199.

[31] Y. Wang, J. Yang, L. Wang, X. Ying, T. Wu, W. An, Y. Guo, Light field image super-resolution using deformable convolution, IEEE Transactions on Image Processing 30 (2021) 1057–1071.

[32] W. Sun, D. Gong, Q. Shi, A. van den Hengel, Y. Zhang, Learning to zoom-in via learning to zoom-out: Real-world super-resolution by generating and adapting degradation, IEEE Transactions on Image Processing In press (2021) 1–16.

[33] X. Wang, K. Yu, C. Dong, C. C. Loy, Recovering realistic texture in image super-resolution by deep spatial feature transform, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, Salt Lake City, Utah, USA.

[34] J. Yoo, N. Ahn, K.-A. Sohn, Rethinking data augmentation for image super-resolution: A comprehensive analysis and a new strategy, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, Online, 2020, pp. 8375–8384.

[35] Y. Xu, S. R. Tseng, Y. Tseng, H. Kuo, Y. Tsai, Unified dynamic convolutional network for super-resolution with variational degradations, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, Online, 2020, pp. 12496–12505.

[36] Y. Mei, Y. Fan, Y. Zhou, L. Huang, T. S. Huang, H. Shi, Image super-resolution with cross-scale non-local attention and exhaustive self-exemplars mining, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, Online, 2020, pp. 5690–5699.

[37] K. Zhang, X. Gao, D. Tao, X. Li, Single image super-resolution with multiscale similarity learning, IEEE Transactions on Neural Networks and Learning Systems, 24 (10) (2013) 1648–1659.

[38] J. Hu, X. Wu, J. Zhou, Noise robust single image super-resolution using a multiscale image pyramid, Signal Processing 148 (2018) 157–171.

[39] M. Bevilacqua, A. Roumy, C. Guillemot, M. L. Alberi Morel, Single-image super-resolution via linear mapping of interpolated self-examples, IEEE Transactions on Image Processing 23 (12) (2014) 5334–5347.

[40] S. Wang, B. Yue, X. Liang, L. Jiao, How does the low-rank matrix decomposition help internal and external learnings for super-resolution, IEEE Transactions on Image Processing 27 (3) (2018) 1086–1099.

[41] K. Zhang, X. Gao, D. Tao, X. Li, Single image super-resolution with non-local means and steering kernel regression, IEEE Transactions on Image Processing, 21 (11) (2012) 4544–4556.

[42] J. Jiang, X. Ma, C. Chen, T. Lu, Z. Wang, Single image super-resolution via locally regularized anchored neighborhood regression and nonlocal means, IEEE Transactions on Multimedia 1 (19) (2017) 15–26.

[43] J. Xu, M. Li, J. Fan, X. Zhao, Z. Chang, Self-learning super-resolution using convolutional principal component analysis and random matching, IEEE Transactions on Multimedia 12 (5) (2019) 1108–1121.

[44] C. Ren, X. He, Y. Pu, Nonlocal similarity modeling and deep cnn gradient prior for super resolution, IEEE Signal Processing Letters 25 (7) (2018) 916–920.

[45] X. Cheng, Z. Fu, J. Yang, Zero-shot image super-resolution with depth guided internal degradation learning, in: European Conference on Computer Vision, Springer, Online, 2020, pp. 265–280.

[46] X. Zeng, H. Huang, C. Qi, Expanding training data for facial image super-resolution, IEEE Transactions on Cybernetics 48 (2) (2018) 716–729.

[47] S. Zhang, X. Li, M. Zong, X. Zhu, R. Wang, Efficient knn classification with different numbers of nearest neighbors, IEEE Transactions on Neural Networks and Learning Systems 29 (5) (2018) 1774–1785.

[48] A. Rahiman V, S. N. George, Robust single image super resolution using neighbor embedding and fusion in wavelet domain, Computers and Electrical Engineering 70 (2018) 674–689.

[49] W. Dong, G. Shi, X. Li, K. Peng, J. Wu, Z. Guo, Color-guided depth recovery via joint local structural and nonlocal low-rank regularization, IEEE Transactions on Multimedia 19 (2) (2017) 293–301.

[50] D. Glasner, S. Bagon, M. Irani, Super-resolution from a single image, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, Kyoto, Japan, 2009, pp. 349–356.

[51] S. Karanwal, M. Diwakar, Od-lbp: Orthogonal difference-local binary pattern for face recognition, Digital Signal Processing 110 (1) (2021) 102948.

[52] M. Haris, G. Shakhnarovich, N. Ukita, Deep back-projection networks for single image super-resolution, IEEE Transactions on Pattern Analysis and Machine Intelligence In press (2020) 1–14.

[53] Y. Zhang, Q. Fan, F. Bao, Y. Liu, C. Zhang, Single-image super-resolution based on rational fractal interpolation, IEEE Transactions on Image Processing 27 (8) (2018) 3782–3797.

[54] C. Dong, C. C. Loy, K. He, X. Tang, Learning a deep convolutional network for image super-resolution, in: European Conference on Computer Vision, Springer, Zurich, Switzerland, 2014, pp. 184–199.

[55] Y. Zhang, Y. Zhang, J. Zhang, Q. Dai, Ccr: Clustering and collaborative representation for fast single image super-resolution, IEEE Transactions on Multimedia 18 (3) (2016) 405–417.

[56] J. Kim, J. K. Lee, K. M. Lee, Accurate image super-resolution using very deep convolutional networks, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, Las Vegas, NV, USA, 2016, pp. 1646–1654.

[57] W. Lai, J. Huang, N. Ahuja, M. Yang, Deep laplacian pyramid networks for fast and accurate super-resolution, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, Honolulu, Hawaii, USA, 2017, pp. 5835–5843.

[58] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, Image quality assessment: From error visibility to structural similarity, IEEE Transactions on Image Processing, 13 (4) (2004) 600–612.

[59] J. W. Soh, S. Cho, N. I. Cho, Meta-transfer learning for zero-shot super-resolution, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, Online, 2020, pp. 3516–3525.

Jian Xu: Conceptualization,Methodology,Software,Funding acquisition
Yan Gao:Formal analysis,Writing - Original Draft,Visualization
Jun Xing:Software
Jiulun Fan:Supervision,Project administration
Qiannan Gao:Software
Shaojie Tang:Investigation

Conflict of interest statement

We declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work, there is no professional or other personal interest of any nature or kind in any product, service or company that could be construed as influencing the position presented in, or the review of, the manuscript entitled “Two-direction Self-learning Super-Resolution Propagation Based on Neighbor Embedding”.