LEARNING APPROACHES FOR FAST IMAGE INTERPOLATION AND SUPER-RESOLUTION

HUANG JUNJIE

M.Phil

The Hong Kong Polytechnic University

2016

THE HONG KONG POLYTECHNIC UNIVERSITY

DEPARTMENT OF ELECTRONIC AND INFORMATION ENGINEERING

Learning Approaches for Fast Image Interpolation and Super-Resolution

Huang Junjie

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Philosophy

August 2015

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____

HUANG JUNJIE

_____

DEDICATION

To my parents

ABSTRACT

Image interpolation and image super-resolution (SR) are fundamental and essential problems in image processing. They share a common purpose which is to enhance the resolution of a low-resolution (LR) digital image, as well as have some dissimilarities. Image interpolation aims to predict missing high-resolution (HR) pixels from the neighboring known ground truth HR pixels, while the objective of image super-resolution is to generate a HR image which preserves sharp edges and natural textures from a blurred and down-sampled version of the original HR image.

The low-quality and small digital images become increasingly undesirable when the resolution of the display screens keeps growing. However, the existing image interpolation and image super-resolution algorithms either produce unsatisfactory HR images with blurry edges and annoying artifacts or require too long processing time to realize practical applications.

We propose a two-stage framework for fast image interpolation via random forests (FIRF). The proposed FIRF method gives high accuracy, as well as requires low computation. The underlying idea of this proposed work is to apply random forests to classify the natural image patch space into numerous subspaces and learn a linear regression model for each subspace to map the LR image patch to HR image patch. The FIRF framework consists of two stages. Stage 1 of the framework removes most of the ringing and aliasing artifacts in the initial Bi-cubic interpolated image, while Stage 2 further refines the Stage 1 interpolated image. By varying the number of decision trees in the random forests and the number of stages applied, the proposed FIRF method can

realize computationally scalable image interpolation. Extensive experimental results show that the proposed FIRF(3,2) method achieves more than 0.3 dB improvement in PSNR over the state-of-the-art Nonlocal Auto-Regressive Modeling (NARM) method. Moreover the proposed FIRF(1,1) obtains similar or better results as NAMR while requires only 0.3% of its computational time.

We also propose to use a set of decision tree strategies for fast and high quality image SR. We take the divide-and-conquer strategy using decision tree for super-resolution (SRDT) which performs a few simple binary tests to classify an input LR patch into one of the leaf nodes and directly multiplies this LR patch with the regression model at that leaf node for regression. Both the classification process and the regression process take extremely small amount of computation. We formulate a hierarchical decision trees (SRHDT) method which cascades multiple layers of super-resolution decision trees to further boost the SR results. Inspired by the random forests approach which combines regression models from an ensemble of decision trees, we propose a hierarchical decision trees approach with fused regression models (SRHDT_f), which fuses the regression models from 4 relevant leaf nodes within the same decision tree to form a more robust regression model. This achieves another 0.1 dB improvement. Our experimental results show that our initial approach, the SRDT method achieves comparable SR results as the sparse representation based method and the deep learning based method but the speed of our method is much faster. Furthermore, our enhanced version, the SRHDT_f method achieves more than 0.3 dB higher PSNR over that of the A+ method which is the state-of-the-art method in SR.

Hybrid DCT-Wiener-Based interpolation scheme using the learnt Wiener filter can significantly improve both objective and subjective performance by learning a suitable Wiener filter to fit the hybrid scheme with a good mix of spatial and transform domain process. Using the adaptive *k*-NN MMSE estimation for each block achieves extraordinary up-sampling results. However, it needs a large database and relatively long processing time. We have also investigated using multiple learnt Wiener filters and combined the information from both external training images and original low-resolution image. The proposed dual MMSE estimators adaptively resolve the problem of one general learnt Wiener filter and use less computation time compared with that of the k-NN MMSE estimation. Experimental results show that the proposed dual MMSE estimators achieve around 1dB PSNR improvement compared to the original hybrid DCT-Wiener-Based scheme and provide more natural visual quality.

LIST OF PUBLICATIONS

JOURNAL PAPERS

[1] **Jun-Jie Huang**, Wan-Chi Siu and Tian-Rui Liu, "Fast Image Interpolation via Random Forests," IEEE Transactions on Image Processing, vol. 24, no. 10, pp. 3232-3245, 2015.

[2] **Jun-Jie Huang** and Wan-Chi Siu, "Learning Hierarchical Decision Trees for Single Image Super-Resolution," paper accepted to be published in IEEE Transactions on Circuits and Systems for Video Technology.

CONFERENCE PAPERS

[3] **Jun-Jie Huang** and Wan-Chi Siu, "Fast Image Interpolation with Decision Tree," Proceedings, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'2015), 19-24 April 2015, Brisbane, Australia.

[4] **Jun-Jie Huang** and Wan-Chi Siu, "Practical Applications of Random Forests for Super-Resolution Imaging," Proceedings, IEEE International Symposium on Circuits and Systems, (ISCAS'2015), 24-27 May 2015, Lisbon, Portugal.

[5] **Jun-Jie Huang** and Wan-Chi Siu, "Large Colour LBP in Generalized Hough Transform", Proceedings, IEEE International Conference on Image Processing, (ICIP'2014), 27-30, October 2014, Paris, France.

[6] **Jun-Jie Huang**, Kwok-Wai Hung and Wan-Chi Siu, "Hybrid DCT-Wiener-Based Interpolation using Dual MMSE Estimator Scheme", Proceedings, 19th International Conference on Digital Signal Processing, (19th DSP2014) 20-23 August 2014, Hong Kong.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my chief supervisor Prof. Wan-Chi Siu for many guidance and supports, which is one of the most important elements to the successful completion of my M.Phil study.

Special thanks are given to Mr. Hui-Wai Lam, Dr. Calvin Cheung, Dr. Yui-Lam Chan, Dr. Ngai-Fong Law and Prof. Kenneth Kin-Man Lam for their invaluable advices and assistance. I am indebted to Dr. Li-Li Wang, Mr. Zhang Chen, Dr. Kwok-Wai Hung, Mr. Hao Wu, Ms. Huiling Zhou, Mr. Meng Yao, Mr. Bochuan Du, Ms. Xuefei Yang and all other member in DSP lab.

Last but not the least, I would like to thank my parents and friends for their love, support and encouragement.

STATEMENTS OF ORIGINALITY

The following contributions reported in this thesis are claimed to be original.

1. Our proposed fast image interpolation via random forests algorithm is the first successful image interpolation method using random forests which achieves the best performance compared to all methods in the literature in terms of image interpolation quality using no more than 2% computation time as the state-of-the-art sparse representation based method. See Chapter 3.

2. For random forests learning, the proposed constraint on split effectively ensures a more balanced decision tree which results in small Mean Squared Error (MSE) on the leaf nodes and achieves higher PSNR. See Chapter 3, Section 3.3.1.5.

3. To make full use of training data, we proposed a two-stage random forests framework which further brings an around 0.2 dB improvement in PSNR compared with single stage random forests scheme. See Chapter 3, Section 3.4.

4. We proposed to use decision tree and hierarchical decision trees for fast and high quality single image super-resolution (SR). We demonstrated that decision tree is suitable and capable of fast image SR, and the proposed image super-resolution using decision tree (SRDT) method achieves comparable SR quality as the state-of-the-art deep learning based method and sparse representation model based method, while it takes less than 1% running time as compared with those two effective approaches. See Chapter 4, Section 4.2.

5. To efficiently and effectively further improve the SR quality, we proposed a hierarchical decision trees framework. This proposed image super-resolution with hierarchical decision trees (SRHDT) method can provide more than 0.4 dB gain in PSNR compared to the SRDT method and achieve best method in the literature. See Chapter 4, Section 4.3.

6. Inspired by random forests approach, we proposed to fuse regression models from relevant leaf nodes in a decision tree to improve the generalization ability of single regression model. The proposed hierarchical decision trees with fused regression model method (SRHDT_f) method provides another around 0.1 dB improvement and achieves more than 0.3 dB PSNR gain over the best results in the literature. See Chapter 4, Section 4.2.2.2 and Section 4.3.

7. With a high temporal correlation in video sequence, we introduced a data dependent model for video SR which can further improve the super-resolved image in a great extent. See Chapter 4, Section 4.4.

8. We proposed a classification based image up-sampling method to improve the Hybrid-DCT-Wiener-Based method. The proposed algorithm adaptively combines the information from both the input image itself and the training images. The proposed dual MMSE estimation scheme can provide more than 1 dB improvement compared with the original Hybrid-DCT-Wiener-Based method with more pleasant up-sampled images. See Chapter 5.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

CNN	Convolutional neural network
DCT	Discrete Cosine Transform
FSIM	Feature Similarity
HR	High-Resolution
IDCT	Inverse discrete cosine transform
k-NN	*k*-nearest neighbor
LR	Low-Resolution
MMSE	Minimum mean square error
MSE	Mean squared error
NE	Neighbor embedding
PSNR	Peak-to-signal noise ratio
SR	Super-resolution
SRM	Sparse representation model
SSIM	Structural Similarity

CHAPTER 1 INTRODUCTION

1.1. IMAGE INTERPOLATION AND IMAGE SUPER-RESOLUTION

Image interpolation and image super-resolution (SR) are algorithms [1-73] which aim to enlarge a small digital image to a large one. Let us call image interpolation and image super-resolution as image up-sampling when we talk about both scenarios. The objective of image up-sampling is to reconstruct a high resolution (HR) image from one or several low resolution (LR) images, while minimizing visual artifacts or minimizing the difference with the original HR image. A pixel in the input LR image is related to multiple pixels in the resultant HR image depending on the upscaling factor p. Thus, image up-sampling is an ill-posed underdetermined inverse problem. There is infinite number of HR image solutions to an input LR image. With $1/p^2$ of the original information (p is the magnification factor), it is difficult to recover the original HR image exactly. The main reason is that the pixel predictions made by image up-sampling algorithms are usually the mean prediction results which may have high variances caused by ambiguity within the LR images.

1.1.1. APPLICATIONS

Despite the difficulty of image up-sampling problem, it has practical significance to break the inherent LR imaging of the devices and to better utilize the growing capability of HR display. The applications of image up-sampling algorithms include face recognition, surveillance system, medical imaging, High Definition Television (HDTV), image coding, image resizing/manipulation, and image compression, etc. For example, in a surveillance system, the captured human faces are usually very small (for instance, 20

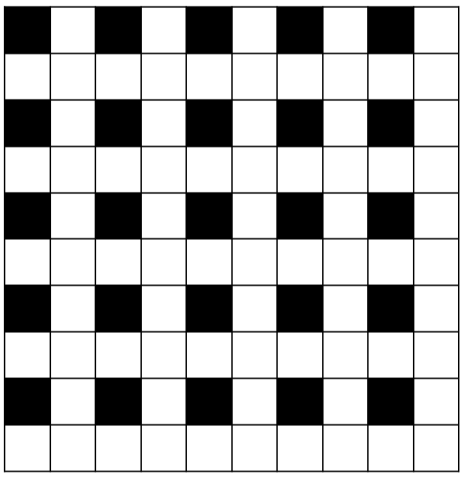


Fig. 1-1: Illustration for image interpolation down-sampling: the black pixels are the ground-truth pixels in LR image obtained from the original HR image, while the white pixels are the unknown HR pixels which are to be interpolated.

pixels × 10 pixels). It is difficult for human beings or for face recognition algorithms to recognize the captured faces. Image up-sampling algorithms can enlarge the interested faces into a larger scale which would be more convenient for human to identify and improve the recognition rate of the face recognition algorithms.

1.1.2. DEGRADATION MODEL

In image interpolation and image SR problem, the vectorized observed LR image patch **y** can be represented as:

$$\mathbf{y} = \mathbf{DHx} + \mathbf{n}, \tag{1-1}$$

where **D** is the down-sampling operator, **H** is the blurring operator, **x** is the vectorized HR image patch, and **n** is the additive noise.

1.1.2.1. IMAGE INTERPOLATION DEGRADATION MODEL

For image interpolation problem, there is no anti-aliasing pre-filtering during degradation and the pixels on LR image are directly down-sampled from the HR image, as shown in Fig. 1-1. Thus, the blurring operator **H** is an identity matrix, the down-

sampling matrix **D** is a sparse matrix which has very few elements with value 1 and most elements with value 0, and there usually is no additive noise. Hence Eq. (1-1) can be reformed into:

$$\mathbf{y} = \mathbf{Dx}, \tag{1-2}$$

where **y** is the vectorized observed LR image patch under the image interpolation assumption, **D** is the down-sampling operator, and **x** is the vectorized HR image patch.

An example of the down-sampling operator **D** when the down-sampling factor is 2 and the HR image patch size is 4 by 4 is:

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{1-3}$$

1.1.2.2. IMAGE SUPER-RESOLUTION DEGRADATION MODEL

For image SR problem, the blurring operator **H** is not a Dirac delta function (identity matrix) as in the image interpolation problem. Most of the image SR algorithms [1-16, 18-36, 38] assume that the blurring operator **H** is a Gaussian blur kernel or a bi-cubic filter kernel. There are also some research works named as blind image super-resolution [74-77] which treat the blurring operator unknown and estimate the blurring operator from cues within the input image itself.

Despite the blind image SR algorithms, there are 3 typical single image SR degradation scenarios, all responding to a zooming deblurring setup with a known blur kernel and zero noise.

(1) A bi-cubic filter followed by down-sampling by a scale of 2.

(2) A bi-cubic filter followed by down-sampling by a scale of 3.

(3) A Gaussian filter of size 7×7 with standard deviation of 1.6 followed by down-sampling by a scale of 3.

1.1.2.2. TRANSFORM DOMAIN IMAGE UP-SAMPLING DEGRADATION MODEL

In image/ video coding, images are divided into non-overlapping blocks which are converted into transform domain (usually Discrete Cosine Transform (DCT) domain). In a DCT block of size $k \times k$, there are one DCT coefficient and k^2-1 AC coefficients. A HR DCT block is down-sampled to a LR DCT block by only retaining the low frequency parts and removing high frequency parts. The obtained LR DCT block is then rescaled and converted back to spatial domain by inverse Discrete Cosine Transform (IDCT).

1.1.3. DIFFERENCE BETWEEN IMAGE INTERPOLATION AND IMAGE SUPER-RESOLUTION

As the image interpolation problem and the image SR problem have different assumptions (degradation models), their algorithms are different in principle.

In image interpolation problem, the pixels on the obtained LR image are directly down-sampled from the original HR image. The observed LR image is very sharp, however, is also highly aliased since there is no anti-aliasing pre-filtering (blur). Thus, we estimate the missing HR pixels based on some ground-truth pixels. The main problem for image interpolation is to insert pixels on the HR image grid between the known pixels and resolve the strong aliasing problem, if any. The missing HR pixels can be roughly predicted based on the statistics of the observed LR pixels.

While in image SR problem, we don't have ground-truth pixels due to the existence of blurring operator which is not an identity matrix. The initially bi-cubic interpolated image is much blurry than the original HR image. For this reason, image SR problem can

be referred to *zooming deblurring*. "*Zooming*" means that the LR image should be enlarged and "*deblurring*" means that the image SR algorithm should be able to restore the sharpness of the super-resolved image.

1.2. LITERATURE REVIEW ON IMAGE INTERPOLATION

Image interpolation techniques can be mainly categorized into three groups: polynomial-based methods [39, 40, 43, 46], edge directed methods [42, 44, 45, 47-50, 52-57] and learning-based methods [51, 58-60].

1.2.1. POLYNOMIAL-BASED

The polynomial-based interpolation methods (i.e. Bi-linear, Bi-cubic, cubic-spline) [39, 40, 43, 46] interpolate the missing pixels using the known neighboring pixels by non-adaptive linear filters. They can achieve real-time performance, however, produce blurry images with ringing artifacts near the edges since non-adaptive linear filters cannot adapt to diverse local pixel appearances.

1.2.2. EDGE-DIRECTED

The edge directed interpolation methods [42, 44, 45, 47-50, 52-57] explicitly or implicitly use the edge direction information adapting to varying local structures and achieve pleasing results which preserve sharp edges. The explicit methods [42, 45, 47] estimate edge directions and then interpolate the missing pixels along the detected edge orientation. Therefore, the performance is limited by edge orientation estimation accuracy. In order to accommodate the drawbacks of the explicit edge-directed interpolation algorithms, implicit methods [44, 48-50, 52-57] have been proposed to statistically estimate the interpolation parameters around the local patch.

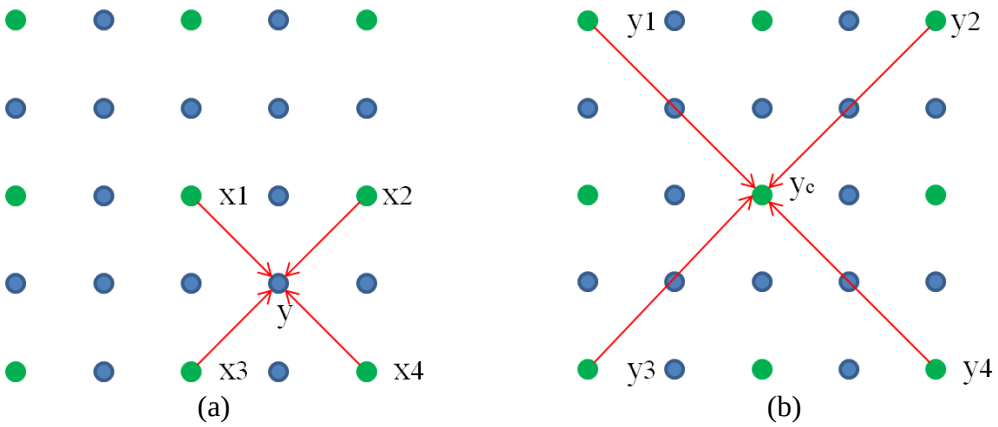


Fig. 1-2: NEDI: the green dots are the observed ground-truth pixels and the blue dots are the pixels to be interpolation. (a) Unknown pixel y to be predicted by four neighboring pixels x_1, x_2, x_3 and x_4. (b) Interpolation weights to be estimated from the observed ground-truth pixels, y_c, y_1, y_2, y_3 and y_4.

The new edge directed interpolation (NEDI) based methods [44, 52-54] estimate HR covariance by LR covariances and interpolate the HR pixels using the estimated covariance based on the geometric duality property which means "the correspondence between the high-resolution covariance and the low-resolution covariance that couple the pair of pixels at different resolution but along the same orientation" [44]. For example, as shown in Fig. 1-2, the covariance between the unknown HR pixel y and the known pixels x_i (i=1,…,4) can be estimated from the covariance between HR pixels y_c and y_i (i=1,…4). The results of NEDI are much sharper than that of the Bi-cubic interpolation, however, with some unnatural patterns which would lead to lower Peak-to-Noise Ratio.

The directional filtering and data fusion (DFDF) method [48] proposed by Zhang and Wu interpolates the missing pixel by combining the two directional estimation (45° and 135°) results using linear minimum mean square error estimation (MMSE). Each directional estimation is considered as a noisy estimation. These directional estimations modeled as different noisy measurements of the missing pixel are fused by the linear

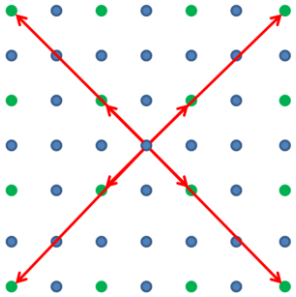


Fig. 1-3: For each pixel to be interpolated, there are two predictions with noise under orthogonal directions. Two prediction pixel values are fused to produce a more robust estimation.

MMSE estimation technique into a more robust estimate, using the statistics of the two observation sets.

The 2-D autoregressive modeling and soft-decision estimation (SAI) method [50, 57] made some modifications to the NEDI method by using block-based soft-decision estimation and increasing order in piecewise autoregressive modeling. Different from NEDI method and DFDF method, the autoregressive model coefficients are optimized for all pixels in a block which is regarded as soft-decision. With a feedback term, the observed LR pixels should be able to be predicted from the estimated HR pixels. This improves the stability of the model. The added horizontal and vertical correlations enhance the prediction power. The SAI method is a representative of the edge-directed image interpolation method and achieves the best performance in a period of time.

The iterative curve based interpolation (ICBI) method [17] uses the local second-order image derivative information to interpolate the missing pixels adaptively and achieved fast image interpolation with pleasing perceptual results.

1.2.3. LEARNING-BASED

The learning-based image interpolation approaches [51, 58-60] try to exploit the relationship between the LR patches and the HR patches from exemplar images or the

image itself to enhance the image details. Among them, the emerging sparse representation model (SRM) has been proven to be effective to image interpolation problem [58, 60]. The SRM-based methods assume that the image patch can be represented by a linear combination of a few columns (atoms) from the learned dictionary. The recently proposed nonlocal autoregressive modeling (NARM) method [58] combines the nonlocal image self-similarity constraint into the sparse representation model. The NARM achieves outstanding results compared to the conventional edge-directed algorithms, however, needs much longer processing time.

1.3. LITERATURE REVIEW ON IMAGE SUPER-RESOLUTION

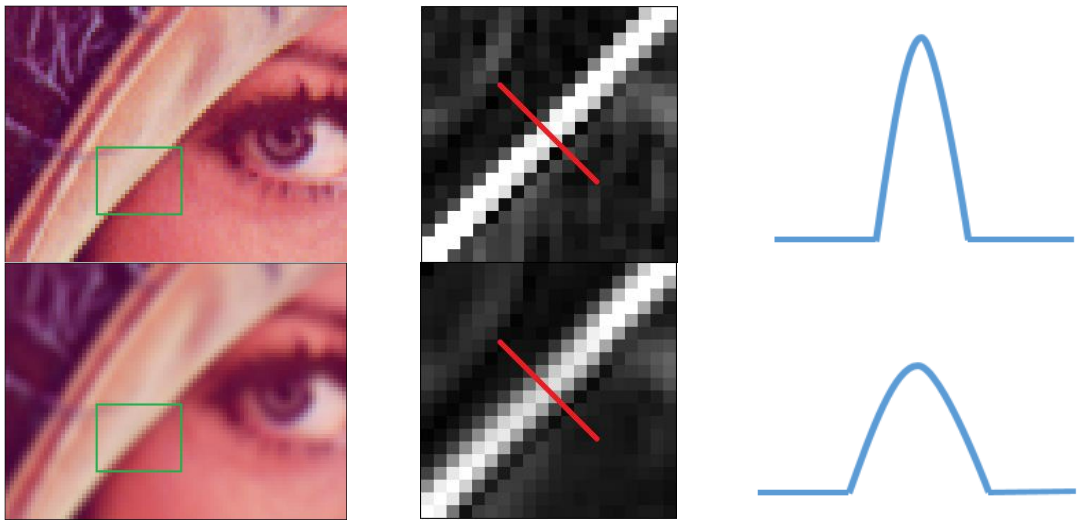
The existing image SR methods in the literature can be classified into three main classes: interpolation-based methods [39, 40, 43, 46], reconstruction-based methods [7, 9-11, 14, 19, 23, 25, 29, 30] and learning-based methods [1-6, 8, 10, 12, 13, 15, 16, 18, 20-22, 24, 26-28, 31, 32, 34-36, 38].

1.3.1. INTERPOLATION-BASED

The interpolation-based algorithms [39, 40, 43, 46] use a based function or an interpolation kernel to estimate the pixels non-adaptively or adaptively. (Please refer back to Section 1.2. LITERATURE REVIEW ON IMAGE INTERPOLATION.)

1.3.2. RECONSTRUCTION-BASED

Reconstruction-based methods [7, 9-11, 14, 19, 23, 25, 29, 30] impose a certain prior knowledge to regularize the ill-posed image SR problem so as to suppress artifacts and reach a solution that is more likely to be the natural image. Some of commonly used natural image priors include total-variation prior [7, 9], gradient-profile prior [19, 29, 30]



(a) edges (b) gradient maps and 1-D trace of gradient profile (c) gradient profile

Fig. 1-4: Gradient profile. (a) Two edges with different sharpness. (b) Gradient maps and 1-D patch perpendicular to the edge direction. (c) Gradient profile.

and nonlocal similarity [11, 14, 23, 25]. Although the reconstruction-based methods can generate sharp edges, the details of the HR image cannot be restored especially for cases with large upscaling factors.

Let us take the gradient-profile prior based method as an example. Sun et. al. [19] proposed to use gradient profile prior for image SR and enhancement. In their method, the image gradients are represented by gradient profiles, which are 1-D profiles of gradient magnitudes perpendicular to image structures as shown in Fig. 1-4. The gradient profiles are modeled by a parametric gradient profile model which is learned from natural image. The image gradient field of the LR image is transformed and guided by the parametric gradient profile model.

1.3.3. LEARNING-BASED

Learning-based approaches [1-6, 8, 10, 12, 13, 15, 16, 18, 20-22, 24, 26-28, 31, 32, 34-36, 38] divide the input LR image into overlapped LR patches and estimate their corresponding HR patches from a set of LR-HR exemplar patch pairs which are cropped

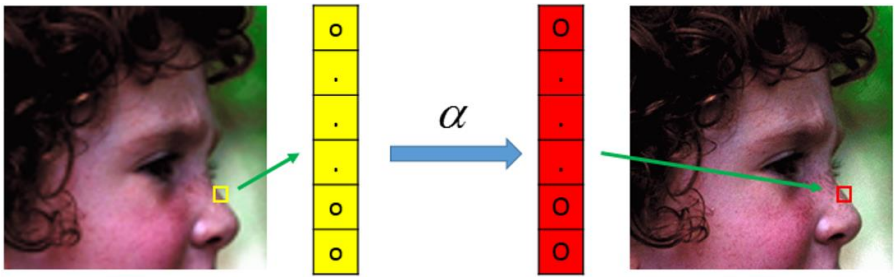


Fig. 1-5: The idea of super-resolution using sparse representation: the left image is the bicubic upscaled LR image and the right image is the super-resolved image, the LR patch and HR patch can be represented by the same sparse signal α with LR and HR dictionary.

from external datasets or the input LR image itself. The essence of learning-based method is to learn an effective and efficient LR-HR patch co-occurrence model for HR patch prediction.

In the pioneering work, entitled as Example-based Super-Resolution [2], Freeman Jones and Pasztor proposed to search *k*-nearest neighbor (*k*-NN) LR-HR patch pairs of the input LR patch from an external dataset and estimate the desired HR patch with the retrieved HR patches in a Markov Random Field framework. Although the super-resolved HR image is much sharper than the Bi-cubic interpolated image, the result appears to be noisy. Chang Yeung and Xiong [3] proposed an image SR method based on Neighbor Embedding (NE) which assumes that the LR patches and their corresponding HR patches share a similar low-dimensional nonlinear manifold. The NE-based approaches [3, 4, 8, 21] have a similar spirit as the Freeman's method [1, 2], however, the predicted HR patch is reconstructed by its weighted *k*-NN where the weights and retrieved *k*-NN are estimated by the input LR patch with LR-HR exemplar patch pairs. The performance of *k*-NN based approaches [1, 2] and NE-based approaches [3-6, 8, 12, 13, 15, 20, 21, 24, 26-28, 31, 34-36, 38] is closely related to the number of the LR-HR exemplar patch pairs. While a huge LR-HR patch pair dataset requires vast memory for storage, it makes the *k*-NN searching time become enormous.

For efficient image SR, a compact representation of the huge number of LR-HR patch pairs turns into necessary. The dictionary-based methods [5, 6, 12, 13, 15, 20, 21, 24, 26-28, 31, 34-36, 38] explicitly or implicitly form a dictionary or coupled LR-HR dictionaries to represent the relationship between LR and HR patches in the training dataset. The Sparse Coding Super-Resolution (ScSR) method [13] proposed by Yang *et al*. jointly learns two coupled over-complete dictionaries based on sparse representation for SR. In the sparse coding framework as shown in Fig. 1-5, each input LR patch can be sparsely represented by a few atoms in the LR dictionary and the desired HR patch is then reconstructed using the same sparse signal with the coupled HR dictionary. Although the NP-hard l_0 norm problem in sparse coding framework has been relaxed to l_1 norm problem, it is still time-consuming to produce the results. Many works [12, 15, 21, 24, 26, 27, 35, 38] have been proposed to improve the ScSR method in recent years. Especially, Zeyde Elad and Protter [24] improved the ScSR method in both runtime efficiency and SR quality by reducing the dimensionality of the feature vectors using the principle component analysis (PCA) and applying the orthogonal matching pursuit for sparse coding. Peleg and Elad [35] suggested clustering the data and cascading several levels of their proposed statistical prediction model which estimates the HR patch sparse representation from the LR patch sparse representation and LR sparsity pattern. This simple "feedforward Neural Network" like inference scheme leads to a low-complexity image SR algorithm.

To further improve the efficiency of the learning-based single image SR, recently some fast SR algorithms [28, 31, 34-36] appeared in the literature. Yang and Yang proposed a simple function method [31] which applies *K*-means clustering method to the training LR patches and uses linear regression to learn a regression model for each cluster.

At runtime, the input LR patch is firstly classified into one of the learned cluster centers and then the regression model belonging to that cluster center is applied for HR patch estimation. Timofte De and Gool proposed the Anchored Neighbor Regression (ANR) method [28], which further relaxes the sparse coding problem from l_1 norm to l_2 norm. This arrangement has a closed form solution learned using the neighborhood dictionary atoms. By dividing the dictionary atoms into K groups, ANR method learns K regression models offline. In the testing phase, each input LR patch feature finds its nearest neighbor dictionary atom by solving the sparse coding problem and then uses the corresponding regression model for prediction. The same research group further improved the ANR method and proposed the adjusted anchored neighbor regression (A+) method [36] which is the state-of-the-art for fast single image SR and achieved the best results in the literature. Instead of using neighborhood dictionary atoms to learn the regression model, the A+ method directly utilizes the dense neighborhood training LR-HR patch pairs of a dictionary atom for regression model learning which offers higher accuracy with the same processing time as the ANR method. Dong *et al*. proposed a super-resolution convolutional neural network (SRCNN) model [34] for single image SR. The SRCNN model consists of three layers of convolutional neural network (CNN) for patch extraction and representation, nonlinear mapping and HR image reconstruction, respectively. The SRCNN method offline learns the convolutional neural network using a huge number of training LR-HR patch pairs with millions of back-propagations and applies the learned SRCNN model during testing.

Most of the emerged fast learning-based SR approaches (Simple Function method [31], ANR method [28] and A+ method [36]) classify the training data into a small number of groups and learn a regression model for each group. However, the classification time

of these methods is linearly related to the number of clusters. It is difficult to make a tradeoff between the image SR quality and the computational time. A more efficient classification method would greatly improve the runtime speed.

1.4. DCT-BASED IMAGE INTERPOLATION

Most image up-sampling schemes are performed in spatial domain. Recent developments show that the Discrete Cosine Transform (DCT)-based image interpolation schemes [61-69, 71-73] can provide better performance and lower complexity.

The conventional DCT-based image resizing algorithms [61, 62, 64-67, 69] which are based on zero-padding approach do not introduce new information into the resized image. Their aim is to resize an image with less computation without significant degradation of the image quality. The algorithm proposed by Dugad and Ahuja [62] directly pads zeros in the high frequency positions of the 8×8 DCT block when up-sampling an image. Mukherjee and Mitra [64] proposed to use subband approximation or truncated approximation when converting DCT coefficients among different sizes of blocks and to convert the 8×8 DCT block into a 16×16 DCT block using zero-padding. The L/M algorithm [65] can resize an image by a rational number L/M, i.e. L-times image up-sampling by zero-padding followed by an M-times image down-sampling by abandoning the high frequency coefficients. Algorithms in [67] and [69] can arbitrary resize an image in the DCT domain by using the spatial relationship of a DCT block and its sub-blocks [63], which helps to bypass the inverse DCT process.

In order to improve the performance of zero-padding based approaches, some hybrid approaches [68, 70-73] were proposed to add new information to the high frequency components. The hybrid DCT-Wiener-Based interpolation scheme [71] combines the

advantages of both DCT domain and spatial domain. In this scheme, the low frequency components are directly copied from the original low-resolution image in the DCT domain and the high frequency components are predicted using a 6-tap Wiener filter [78] in the spatial domain. Consequently, the low frequency part retains to have good fidelity and the high frequency part can be better than those in zero-padding approach since the spatial correlation can help to recover part of the high frequency. Cho and Lee [68] proposed to use the correlation between the low frequency coefficients and the high frequency coefficients in the DCT domain to predict missing high frequency coefficients. A MMSE estimator is used for the prediction, while if the training image and the testing image are less correlated, the PSNR improvement is limited. Hung and Siu [72] proposed to train the Wiener filter using MMSE estimation in the spatial domain with a block-based structure because they found that the 6-tap Wiener filter is not suitable for the DCT-WB scheme since there is a one quarter-pixel shift between the pixel positions of the 6-tap Wiener filter up-sampled image and the original high-resolution image. By using the learnt Wiener filter in the DCT-WB scheme, about 0.4 dB PSNR improvement is achieved. In [72] a non-adaptive Wiener filter is trained for all blocks, Hung and Siu [73] recently proposed using an online learning adaptive Wiener filter for each block in the testing image to further improve the performance. The adaptive Wiener filter is trained using several hundreds of nearest neighbor blocks of the testing blocks selected among a large database. As a result, each trained Wiener filter will be very suitable for that particular block in the testing image. More than 1dB PSNR improvement was achieved in reference [71].

1.5. ORGANIZATION OF THESIS

The rest of the thesis is organized as follows. In Chapter 2, we make a technical review for basic image interpolation and image SR techniques. In Chapter 3, we propose a two-stage framework for fast image interpolation via random forests which achieved state-of-the-art image interpolation quality while only takes the sparse representation based method's 0.3% computational time. In Chapter 4, we propose to use a set of decision tree strategies for fast and high quality image SR. Our initial approach, the SRDT method achieves comparable SR results as the sparse representation based method and the deep learning based method but the speed of our method is much faster. Furthermore, our enhanced version, the SRHDT_f method which applies fused regression models for more accurate HR patch prediction achieves more than 0.3 dB higher PSNR over that of the A+ method which is state-of-the-art method in SR. In Chapter 5, we improve the Hybrid DCT-Wiener-Based interpolation scheme and propose to use multiple learnt Wiener filters and combine the information from both external training set of images and the original low-resolution image. In Chapter 6, the whole thesis is concluded and future research directions are discussed.

CHAPTER 2 TECHNICAL REVIEW

2.1. REGRESSION

The learning-based image interpolation algorithms [51, 58, 59] and image SR algorithms [12, 13, 15, 18, 20-22, 24, 26-28, 31, 32, 35, 36, 38] are usually modeled as a regression problem which is learned using supervised learning. Regression is a statistical process to estimate the relationship between input and output variables which are continuous. The supervised learning problem [79] is to find the model parameters which could balance the fitting error between the estimated values and the ground-truth values and its generalization ability. The training data for image interpolation/SR problem are in the form of LR-HR patch pairs $\{(\mathbf{x}, \mathbf{y})\}$, where $\mathbf{x}$ is the vectorized LR patch and $\mathbf{y}$ is the vectorized HR patch (in this thesis all vectors are column vectors). A typical objective function of the supervised learning is shown below:

$$\boldsymbol{\omega}^* = \arg\min_{\boldsymbol{\omega}} \sum_i L(\mathbf{y}_i, f(\mathbf{x}_i, \boldsymbol{\omega})) + \lambda\Omega(\boldsymbol{\omega}), \tag{2-1}$$

where $\boldsymbol{\omega}$ is the model parameters, $\mathbf{x}_i$ is the observed value, $\mathbf{y}_i$ is the ground-truth value, $f(,)$ is the regression model, $f(\mathbf{x}_i, \boldsymbol{\omega})$ is the fitting value, $L(,)$ is the loss function, λ is the regularization parameter, and $\Omega(\boldsymbol{\omega})$ is the regularization term.

The loss function $L(,)$ is used to evaluate the fidelity between ground-truth values and fitting values. Some typical loss functions include squared loss (for least square), hinge loss (for Support Vector Machine [80]), exponential loss (for boosting [81]) and logarithm loss (for logistic regression [79]), etc. The regularization term can be used to restrict the model property (such as smoothness, sparsity, low-rank, etc) and merge the prior

knowledge into the model. The commonly used regularization term include l_0 norm, l_1 norm, l_2 norm and nuclear norm, etc.

2.1.1. LINEAR REGRESSION

Depending on different choices of the regression model $f(,)$, the regression function can be linear and nonlinear. Many real processes are nonlinear, however, can be approximated with linear models or a combination of piecewise linear models. The advantage of linear model is that it can be solved analytically with a closed form solution.

In general, a linear model can be written as below:

$$y_j = f(\mathbf{x}, \boldsymbol{\omega}) = \sum_{i=1}^{N} \omega_{ij} x_i. \tag{2-2}$$

Or in matrix form:

$$\mathbf{Y} = \boldsymbol{\omega}\mathbf{X}. \tag{2-3}$$

Without considering the regularization term, the objective function in Eq. (2-1) minimizes the fitting error. If the loss function $L(,)$ is a squared loss:

$$\boldsymbol{\omega}^* = \arg\min_{\boldsymbol{\omega}} \| \mathbf{Y} - \boldsymbol{\omega}\mathbf{X} \|_2^2 . \tag{2-4}$$

The cost function which is the square error is formulated as:

$$J(\boldsymbol{\omega}) = (\mathbf{Y} - \boldsymbol{\omega}\mathbf{X})^{\mathrm{T}} (\mathbf{Y} - \boldsymbol{\omega}\mathbf{X}). \tag{2-5}$$

Take the derivative of the cost function:

$$\begin{aligned} \frac{\partial J(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}} &= \frac{\partial}{\partial \boldsymbol{\omega}} (\mathbf{Y} - \boldsymbol{\omega}\mathbf{X})^{\mathrm{T}} (\mathbf{Y} - \boldsymbol{\omega}\mathbf{X}) \\ &= \frac{\partial}{\partial \boldsymbol{\omega}} [\mathbf{Y}^{\mathrm{T}}\mathbf{Y} - \mathbf{X}^{\mathrm{T}}\boldsymbol{\omega}^{\mathrm{T}}\mathbf{Y} - \mathbf{Y}^{\mathrm{T}}\boldsymbol{\omega}\mathbf{X} + \mathbf{X}^{\mathrm{T}}\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\omega}\mathbf{X}] \\ &= -2\mathbf{Y}\mathbf{X}^{\mathrm{T}} + 2\boldsymbol{\omega}\mathbf{X}\mathbf{X}^{\mathrm{T}}. \end{aligned} \tag{2-6}$$

In order to obtain the minimum error, we make the derivative of the objective function equal to zero.

$$2\mathbf{Y}\mathbf{X}^{\mathrm{T}} = 2\boldsymbol{\omega}\mathbf{X}\mathbf{X}^{\mathrm{T}}. \tag{2-7}$$

If $\mathbf{X}\mathbf{X}^{\mathrm{T}}$ is invertible:

$$\boldsymbol{\omega} = \mathbf{Y}\mathbf{X}^{\mathrm{T}}(\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}. \tag{2-8}$$

Eq.(2-8) shows the closed form solution for a linear regression problem. However, when the number of samples is smaller than the dimension of the data, the matrix $\mathbf{X}\mathbf{X}^{\mathrm{T}}$ will not be full-rank and not invertible. Thus, we cannot get the inverse of $\mathbf{X}\mathbf{X}^{\mathrm{T}}$. Or, we have infinite number of solutions (overfitting).

When the regularization term $\Omega(\boldsymbol{\omega})$ is a l_2 norm, the linear regression is called the ridge regression. The l_2 norm regularization term tries to make every element of the model parameter very small (approaching zero) and leads to more robust and stable solution.

$$\boldsymbol{\omega}^{*} = \arg\min_{\boldsymbol{\omega}} \| \mathbf{Y} - \boldsymbol{\omega}\mathbf{X} \|_2^2 + \lambda^2 \| \boldsymbol{\omega} \|_2^2 . \tag{2-9}$$

The regularized quadratic cost function:

$$J(\boldsymbol{\omega}) = (\mathbf{Y} - \boldsymbol{\omega}\mathbf{X})^{\mathrm{T}}(\mathbf{Y} - \boldsymbol{\omega}\mathbf{X}) + \lambda^2 \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\omega}. \tag{2-10}$$

Similar to the derivations in Eq. (2-6):

$$\begin{aligned} \frac{\partial J(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}} &= \frac{\partial}{\partial \boldsymbol{\omega}}(\mathbf{Y} - \boldsymbol{\omega}\mathbf{X})^{\mathrm{T}}(\mathbf{Y} - \boldsymbol{\omega}\mathbf{X}) + \lambda^2 \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\omega} \\ &= \frac{\partial}{\partial \boldsymbol{\omega}}[\mathbf{Y}^{\mathrm{T}}\mathbf{Y} - \mathbf{X}^{\mathrm{T}}\boldsymbol{\omega}^{\mathrm{T}}\mathbf{Y} - \mathbf{Y}^{\mathrm{T}}\boldsymbol{\omega}\mathbf{X} + \mathbf{X}^{\mathrm{T}}\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\omega}\mathbf{X} + \lambda^2 \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\omega}] \\ &= -2\mathbf{Y}\mathbf{X}^{\mathrm{T}} + 2\boldsymbol{\omega}\mathbf{X}\mathbf{X}^{\mathrm{T}} + 2\lambda^2 \mathbf{I}\boldsymbol{\omega}. \end{aligned} \tag{2-11}$$

Let us make the derivative equal to zero.

$$2\mathbf{Y}\mathbf{X}^{\mathrm{T}} = 2\boldsymbol{\omega}\mathbf{X}\mathbf{X}^{\mathrm{T}} + 2\lambda^2 \mathbf{I}\boldsymbol{\omega}. \tag{2-12}$$

From the learning perspective, l_2 norm can prevent overfitting and increase generalization ability. From optimization perspective, l_2 norm can help to improve the

condition number for matrix inversion problem (make the ill-condition problem become well-condition). The regularization term makes ($\mathbf{X}\mathbf{X}^{\mathrm{T}}+\lambda^2\mathbf{I}$) invertible.

$$\boldsymbol{\omega} = \mathbf{Y}\mathbf{X}^{\mathrm{T}}(\mathbf{X}\mathbf{X}^{\mathrm{T}} + \lambda^2\mathbf{I})^{-1}. \tag{2-13}$$

2.1.2. SPARSITY

Let **D** be an over-completed dictionary (matrix) with *K* atoms (columns) and *n* rows and $K > n$. Every column of **D** is a building block for a signal. If a signal **x** can be sparsely represented by a linear combination of a few atoms of **D**, then **α** is the sparse signal of **x** with respect to **D** and only has a few nonzero elements.

$$\mathbf{x} \approx \mathbf{D}\boldsymbol{\alpha}. \tag{2-14}$$

The sparse decomposition problem can be stated as:

$$\arg\min_{\boldsymbol{\alpha}} \frac{1}{2} \| \mathbf{x} - \mathbf{D}\boldsymbol{\alpha} \|_2^2 + \lambda\psi(\boldsymbol{\alpha}), \tag{2-15}$$

where $\psi(\boldsymbol{\alpha})$ induces the sparsity of **α**, and λ is the regularization parameter to balance the fidelity term and regularization term.

Eq. (2-15) prefers a sparser sparse coding **α** (most of its coefficients are zeros) which could provide high fidelity to the input signal **x** with respect to a known dictionary **D**. The sparsity of the sparse signal is represented by the l_0 norm (counting number of non-zero coefficients). However, it is NP-hard to find solution for the l_0 norm problem. The optimal solution can only be obtained by greed search.

$$\psi(\boldsymbol{\alpha}) = \| \boldsymbol{\alpha} \|_0 = \#\{ i \text{ s.t.} \boldsymbol{\alpha}[i] \neq 0 \} \tag{2-16}$$

The l_1 norm is a convex relaxation of the l_0 norm.

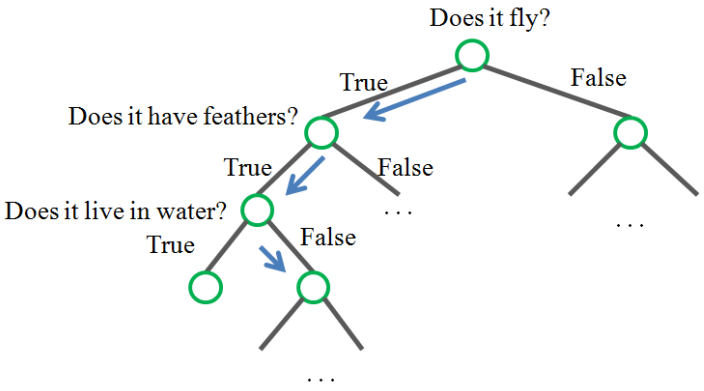


Fig. 2-1: An example which applies decision tree to identify an unknown animal's name.

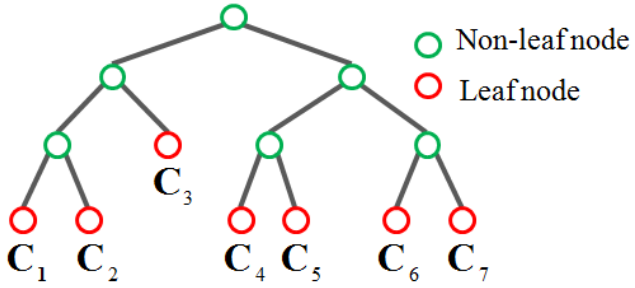


Fig. 2-2: A binary decision tree: the green circles are non-leaf nodes, the red circles are leaf nodes, and each leaf node has a prediction model **C**.

$$\psi(\boldsymbol{\alpha}) = \| \boldsymbol{\alpha} \|_1 = \sum_{i=1}^{p} | \boldsymbol{\alpha}[i] |. \tag{2-17}$$

The sparsity of $\boldsymbol{\alpha}$ is then usually represented by l_1 norm. The reason that l_1 norm can induce sparsity in that the gradient of l_1 norm is constant while the gradient of l_2 norm goes to vanish when the sparse coefficient approaches 0. Thus, a l_1 norm regularization term would make most of the coefficients in $\boldsymbol{\alpha}$ zero, and a l_2 norm regularization term only facilitates small coefficients in $\boldsymbol{\alpha}$.

2.2. DECISION TREE

Decision tree was firstly proposed by Breiman *et al*. [82] in 1984, and is now a commonly used data mining algorithm. The idea of decision tree is to solve a complex problem by testing some simple questions which are organized hierarchically, partitioning

the problem to a more specific region of the decision space according to the answers to the questions and making a decision when the problem reaches a region where the response is confident enough. For example as shown in Fig. 2-1, we can identify the name of unknown animal by asking a series of questions, such as whether it flies, has feathers, lives in water, eats fish, etc. When the number of candidate animals fulfilling the conditions reduces to one, you could identify this unknown animal.

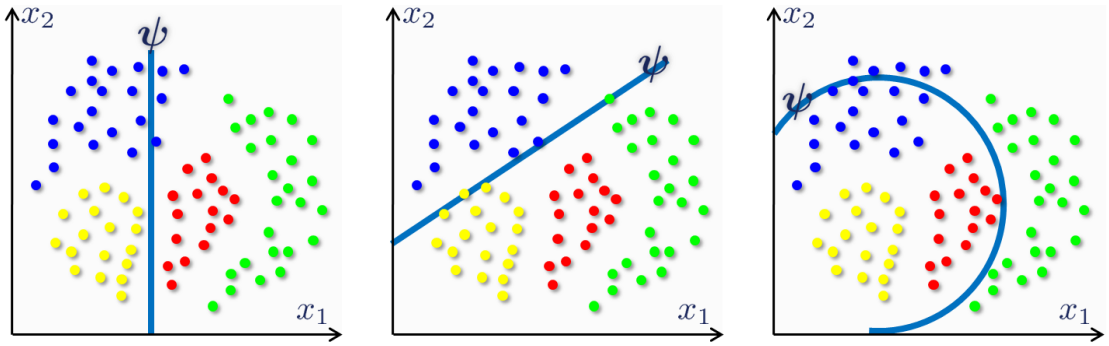
A binary decision tree is in a tree-structure where a node with two child nodes is called a non-leaf node and a node without a child node is called a leaf node. A non-leaf node is responsible for classification by asking a simple question and partitioning the training or the testing data into its left or right child node according to the answer to the question. At each leaf node, a prediction model (a classifier or a regressor) is learned using the arrived training data. The testing data are mapped to its desired form with the prediction models.

2.2.1. BINARY TESTS AND SPLIT FUNCTIONS

Each non-leaf node associates a binary test θ which specifies the parameters of a testing condition to partition the training or the testing data. While the split function $h(\mathbf{x}, \theta)$ defines the testing condition and has binary outputs $\{0, 1\}$ (represents "true and false" or "left and right"). The split function relates to the decision boundary type of the data separation including linear separation and nonlinear separation.

2.2.1.1. LINEAR DATA SEPARATION

There are two types of linear data separation split functions: axis aligned hyperplane (an example is shown in Fig. 2-3 (a)) and general oriented hyperplane (an example is



(a) Axis aligned (b) General oriented (c) Quadratic surface

Fig. 2-3: Three typical split functions: (a) axis aligned hyperplane, (b) general oriented hyperplane, and (c) quadratic surface.

shown in Fig. 2-3 (b)). The general oriented hyperplane is the generalized version of axis aligned hyperplane. The corresponding binary test θ of the general oriented hyperplane is defined as:

$$\theta = (\phi, \boldsymbol{\psi}, \tau), \tag{2-18}$$

where ϕ=ϕ(**x**) selects n (n > 1) features from the input feature vector **x**, $\boldsymbol{\psi}$ is a $n \times 1$ matrix and denotes the coefficients for a generic line in homogeneous coordinates, and τ is the threshold value.

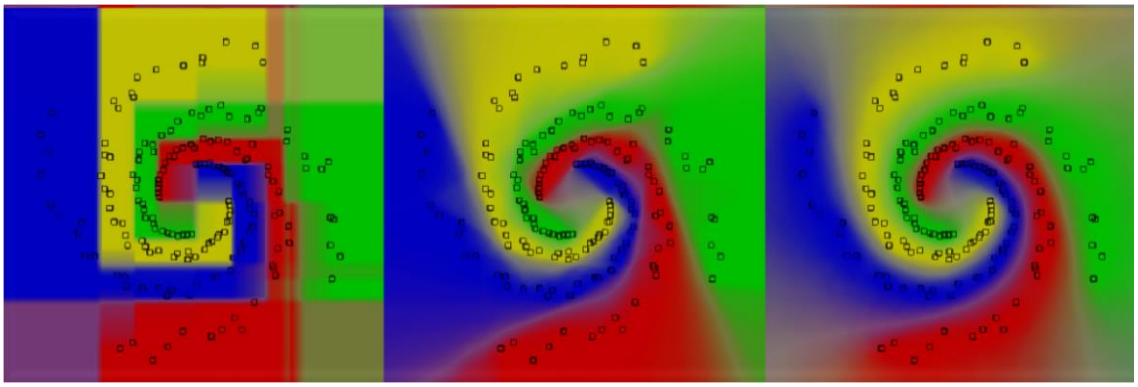
The binary test parameters can describe a hyperplane which can separate the affine space into two half-space.

$$\phi(\mathbf{x}) \cdot \boldsymbol{\psi} = \tau \Leftrightarrow \psi_1 x'_1 + \psi_2 x'_2 + \ldots + \psi_n x'_n = \tau. \tag{2-19}$$

The split function determines which half-space the input feature vector **x** belongs to and is defined as:

$$h(\mathbf{x}, \theta) = \begin{cases} 0, & \text{if } \phi(\mathbf{x}) \cdot \boldsymbol{\psi} < \tau \\ 1, & \text{otherwise.} \end{cases} \tag{2-20}$$

For axis aligned hyperplane, ϕ only select one feature (n = 1) from the input feature vector **x**. The defined hyperplane is perpendicular to the selected feature axis. The general



(a) Axis aligned (b) General oriented (c) Quadratic surface

Fig. 2-4: Classification results by different binary tests and split functions: (a) axis aligned hyperplane, (b) general oriented hyperplane, and (c) quadratic surface.

oriented hyperplane has higher degree of freedom over the axis aligned hyperplane. It is seen from Fig. 2-4 that the general oriented hyperplane can make the classification boundary adhere to the inherent structure in the data space, which is better than the axis aligned hyperplane.

2.2.1.2. NONLINEAR DATA SEPARATION

A more complex binary test could be beneficial especially when the data space is linearly inseparable. In Fig. 2-4 (c), the classification result by a quadratic surface shows better generalization property than the linear data separation binary tests and split functions (Fig. 2-4 (a) and (b)). The nonlinear data separation is achieved by replacing the hyperplane with a hyper-surface (for example, a conic section as shown in Fig. 2-3 (c)). The binary test θ for a conic section is defined as:

$$\theta = (\phi, \boldsymbol{\psi}, \tau), \tag{2-21}$$

where $\phi=\phi(\mathbf{x})$ which selects n ($n > 1$) features from the input feature vector $\mathbf{x}$, $\boldsymbol{\psi}$ is a $n \times n$ matrix and denotes the conic section in homogeneous coordinates, and τ is the threshold value.

The conic section defines the boundary which is used to separate the data space.

$$\boldsymbol{\psi} \cdot \phi(\mathbf{x}) \cdot \boldsymbol{\psi} = \tau. \tag{2-22}$$

The corresponding split function is defined as:

$$h(\mathbf{x}, \theta) = \begin{cases} 0, & \text{if } \boldsymbol{\psi}^{\mathrm{T}} \cdot \phi(\mathbf{x}) \cdot \boldsymbol{\psi} < \tau \\ 1, & \text{otherwise.} \end{cases} \tag{2-23}$$

One can select more complex split functions such as SVM [83] and neural networks [84], etc. Higher discrimination power could be obtained; however, the complexity of the binary test parameters should also go explored. However, the training time may become much longer (for detailed explanations please refer to Section 2.2.4.1. Optimal Binary Tests).

2.2.2. TRAINING SETS

For a regression problem, the training data is in the form of observed data **x** and ground-truth data **y** pairs (e.g. the LR-HR patch pairs in image up-sampling problem). Let us denote a pair of training data as $\{P_i = (\mathbf{x}_i, \mathbf{y}_i)\}$ and assume that there are *l* pairs of training data. The whole training dataset is denoted as $S_0 = \{P_i \mid i = 1, \ldots, l\}$. In supervised learning algorithms, the quantity and the quality of the training data could greatly affect the training results. It comes to the stage that learning algorithms themselves are very simple and the complexity of a learning algorithm comes from the data. Thus, the training data should be carefully selected during preparation.

2.2.3. ENERGY MODELS

During training, a set of binary tests will be evaluated and the best one will be applied to split the training data and stored for testing. The goodness of each binary test is measured by energy models, including entropy and information gain.

2.2.3.1. ENTROPY

In information theory, entropy measures the average information (uncertainty) in the data. The decision tree gradually partitions the training data into a more specific region by the binary tests on the non-leaf nodes. This corresponds to entropy decrease for the training data. The training data at the deeper nodes in the decision tree have smaller entropy. Thus, one of objectives for decision tree training is to find good binary tests which can achieve the lowest entropy among all the candidate binary tests.

For categorical and non-parametric distribution, the Shannon entropy is usually adopted:

$$H(S) = -\sum_{c \in C} p(c) \log(p(c)), \tag{2-24}$$

where S is the training data at that node, c is the class label, C is the set of all classes, and $p(c)$ is the distribution probability of class c within the training data S which could be easily estimated.

For continuous and parametric distribution, differential entropy is one option:

$$H(S) = -\int_{\mathbf{y} \in Y} p(\mathbf{y}) \log(p(\mathbf{y})) d\mathbf{y}, \tag{2-25}$$

where $\mathbf{y}$ is a continuous label, Y is the value range of $\mathbf{y}$, and $p(\mathbf{y})$ is the probability density of $\mathbf{y}$ within the training dataset S.

When the probability density $p(\mathbf{y})$ is modeled by a Gaussian model $N(\mu, \sigma^2)$, the differential entropy can be derived analytically as:

$$
\begin{aligned}
H(S) &= -\int_{\mathbf{y}\in Y} p(\mathbf{y}) \log(\, p(\mathbf{y}))d\mathbf{y} \\
&= -\int_{\mathbf{y}\in Y} p(\mathbf{y}) \log(\frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(y-\mu)^2}{2\sigma^2}))d\mathbf{y} \\
&= -\int_{\mathbf{y}\in Y} p(\mathbf{y}) (\log(\frac{1}{\sqrt{2\pi\sigma^2}}) - \frac{(y-\mu)^2}{2\sigma^2})d\mathbf{y} \\
&= \frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2} E[(y-\mu)^2] \\
&= \frac{1}{2}\log(2\pi e\sigma^2).
\end{aligned}
\tag{2-26}
$$

If we assume this model to be a *d*-variate Gaussian, the differential entropy can be formulated as:

$$
H(S) = \frac{1}{2}\log((2\pi e)^d \mid \Lambda(S) \mid), \tag{2-27}
$$

where Λ(S) is the covariance matrix of *S*.

Then entropy models stated in Eq. (2-24) and Eq. (2-25) are only related to the variance σ^2 and covariance matrix Λ(S) which are computed from the probabilistic line fitting. The objective of decision tree training turns out to minimize the variance evaluated at the training data. Thus, the decision tree training tries to reduce the ambiguity among the training data and makes each training datum have a confident (i.e. with low variance) fitting value.

Different from the differential entropy, the entropy of a continuous distribution can be modeled as the fitting error of the training data.

$$
H(S) = \sum_{P_i \in S} \| \mathbf{y} - f(\mathbf{x}, \boldsymbol{\omega}) \|_2^2, \tag{2-28}
$$

where $f(\mathbf{x}, \boldsymbol{\omega})$ is the fitting value.

This approach has similar reasoning as the differential entropy approach. Because the fitting value $f(\mathbf{x}, \boldsymbol{\omega})$ is the average prediction value and the mean squared loss is related to

variance of the estimated value. The difference is that using fitting error to evaluate entropy is more direct than differential entropy and is without the constraint of Gaussian assumption. The regression model $f(\mathbf{x}, \boldsymbol{\omega})$ is usually the same as the leaf prediction model stated in Section 2.2.4.3.

2.2.3.2. INFORMATION GAIN

Information gain $I(S, \theta)$ is applied to measure the entropy (uncertainty) reduction achieved by partition the training data S into left and right child nodes S^L and S^R using a binary test θ.

$$\begin{aligned} S^L &= \{P_k | h(\mathbf{x}_k, \theta) = 0, P_k \in S\} \\ S^R &= S \setminus S^L \end{aligned} . \tag{2-29}$$

$$I(S,\theta) = H(S) - \sum_{n \in \{L.R\}} \frac{|S^n|}{|S|} H(S^n), \tag{2-30}$$

where $|S|$ is defined as the cardinality of training data S.

The information gain could be understood as the difference between the entropy at the parent node and the weighted entropy at the child nodes.

2.2.4. TREE TRAINING

During training, the root node is initialized by all the training data S_0 and considered as an unprocessed non-leaf node. The aim of tree training is to construct a decision tree which could effectively classify training data into leaf nodes through hierarchical binary tests at the non-leaf nodes and make predictions with high generalization ability at leaf nodes. The three key elements of a decision tree to be learned during training are (1) optimal binary tests stored at the non-leaf nodes; (2) construction of the tree-structure; and (3) prediction model at the leaf nodes.

2.2.4.1. OPTIMAL BINARY TESTS

For each unprocessed non-leaf node, a set of K candidate binary tests $\Theta = \{\theta_i \mid i = 1,..., K\}$ will be generated to try to split the training data at this node. The corresponding information gain of each test is evaluated. The binary test which achieves the highest information gain will be selected as the optimal binary test θ^* at this non-leaf node.

$$\theta^* = \arg\max_{\theta_i \in \Theta} I(S, \theta_i). \tag{2-31}$$

The number of K determines the randomness during splitting the decision tree nodes and controls the training speed to construct the decision tree. A small K introduces higher randomness between decision trees and little tree correlation. A large K decreases the randomness and increases tree correlation.

In theory, K could be chosen as the number of all possible tests of the discrete variables, i.e. searching over the whole parameter space. The best binary tests among all possible binary test will be selected at the non-leaf nodes and there is no randomness during training. For a single decision tree, maximal K could be beneficial for the quality of training result. However, for random forests (which is an ensemble of decision trees and will be discussed in Section 2.3) without bagging, choosing the maximal value of K will make every decision tree the same and highly correlated which diminishes the better generalization advantage of random forests. Besides, this strategy has very low efficiency, since the size of the parameter space could go extremely large when the dimensionality of the feature vector is large or the binary test parameters are very complex.

Randomized node optimization (RNO) is a more efficient approach which randomly generates a subset of all the possible binary tests. The extremely randomized trees [85]

suggested that the default values of K should be $\sqrt{d}$ and d for classification problem and regression problem respectively, where d is the number of attributes of the observed feature vector. However, the suggested default values may not be the most suitable values for certain applications. One can choose an appropriate value of K through cross-validation adapting to a particular requirement.

2.2.4.2. TREE-STRUCTURE CONSTRUCTION

When the optimal binary test is found, two child nodes of this non-leaf node will be constructed and the training data at this non-leaf node will be exclusively partitioned into its child nodes according to the optimal binary test as in Eq. (2-29).

There are three circumstances that a non-leaf node will not construct its child nodes and not partition its training data. (1) This non-leaf node reaches the pre-defined maximum tree depth, D_{max}. (2) The information gain achieved by the optimal binary test is smaller than a threshold, I_T. (3) The number of training data at this non-leaf node is too few to learn a robust predication model. If any of these three conditions is met, this non-leaf node will be declared as a leaf node.

The maximum tree depth D_{max} restricts the decision tree size. The testing time of a decision tree is strongly related to D_{max} (or average number of non-leaf nodes (binary tests) a testing datum will go through). The information gain threshold I_T ensures the quality of the decision tree. The negative information gain has no positive impact on the testing data prediction accuracy. Sometimes, the decision tree complexity (decision tree depth) is inserted into the information gain formulation which makes a tradeoff between the decision tree size and the information reduction. The minimum number of training data at a leaf node is closely connected with the testing prediction accuracy of the prediction

model constructed by the training data at that node. When the number of training data is insufficient, the generalization ability would be severely declined. Under this consideration, a node will be declared as a leaf node when the training data are too few to split.

2.2.4.3. LEAF PREDICTION MODELS

Non-leaf node is responsible for classification, while leaf node is to make the prediction. The non-leaf node splits the data into its left or right child node according to the result of the split function with a binary test. When a testing data reaches a leaf node, a prediction should be made with reference to the subset of training data at that node. It is assumed that the testing data reached have similar statistics of the training data reached. Thus, the desired value of a testing data is predictable.

For a classification problem, the predicted class label should be the class label that has the highest probability within the training data at that leaf node. For a regression problem, any regression model could be applied, such as linear regression model, Gaussian regression model, neural networks, etc.

2.2.6. TREE TESTING

The testing algorithm of decision tree is very simple. The testing data **x** is passed to the learned decision tree. Starting from the root node, the testing data **x** is tested by the optimal binary tests on the non-leaf nodes and partitioned to the left or right child node until it reaches a leaf node. The prediction model (a classifier or a regressor) at the leaf node estimates the output value (a class label for a classification problem or a continuous value for a regression problem) for the input testing data **x**.

2.3. RANDOM FORESTS

The random forests approach [86] is an ensemble of decision trees which improves the generalization ability. The key idea is to combine prediction models from de-correlated decision trees to reduce prediction variance. The reason behind is that each prediction result by a single decision tree could have low bias and high variance; if all the decision trees are de-correlated, their average prediction results may achieve low bias and low variance.

2.3.1. BAGGING

Breiman [86] proposed the bagged training which is a method to reduce overfitting and improve generalization capability. Each decision tree in the random forests is trained using a different randomly sampled subset of the whole training dataset. Bagging helps to prevent the decision trees overfitting to a particular training dataset and insert extra randomness into the decision trees which makes the decision trees de-correlated. Although bagging could accelerate the training speed (as only a subset of the training data is applied for training), the training efficiency is decreased because not all training data is utilized for all decision trees.

2.3.2. COMBINING DECISION TREES

During testing, a testing data **x** is passed to all the decision trees in the random forests. The testing data **x** will retrieve a prediction model from the reached leaf node for each decision tree. The prediction retrieving procedure could be done on parallel which makes random forests algorithm be able to achieve high runtime efficiency, because decision trees are independent from each other. The prediction models from all the decision trees

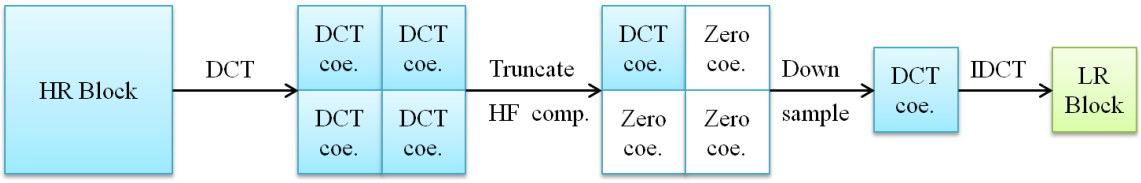


Fig. 2-5: DCT block down-sampling.

in the random forests are usually averaged [86] to form a more robust model.

2.4. DCT-BASED INTERPOLATION

In this section, we review DCT-based down-sampling, DCT-based up-sampling based on zero-padding [62] and DCT-based up-sampling using hybrid DCT-Wiener-based approach [71].

2.4.1. DCT-BASED DOWN-SAMPLING

During DCT-based down-sampling, a 2-D HR image patch $\{x(m, n), 0 \le m \le N\text{-}1, 0 \le n \le N\text{-}1\}$ is firstly converted to DCT domain.

$$C(k,l) = \frac{2}{N}\alpha(k)\alpha(l)\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}(x(m,n)\cos(\frac{(2m+1)\pi k}{2N})\cos(\frac{(2n+1)\pi l}{2N})), \qquad (2\text{-}32)$$

where $0 \le k, l \le N\text{-}1$ and $\alpha(p) = \begin{cases} \sqrt{\frac{1}{2}} & \text{for } p = 0 \\ 1 & \text{otherwise} \end{cases}$.

Let the low frequency part of a 2-D DCT block be $C(k,l)$ with $0 \le k \le K-1$, $0 \le l \le L-1$. For dyadic down-sampling, we assume $K = L = N/2$ which means the down-sampled image patch has half size of the original HR image patch. To truncate the high-frequency part of the DCT block, only the low frequency part is retained and approximated by truncated approximation [64].

$$\overline{C}(k,l) = \begin{cases} 2C(k,l), & k,l = 0,1,\ldots,\frac{N}{2}-1 \\ 0, & \text{otherwise} \end{cases}. \qquad (2\text{-}33)$$

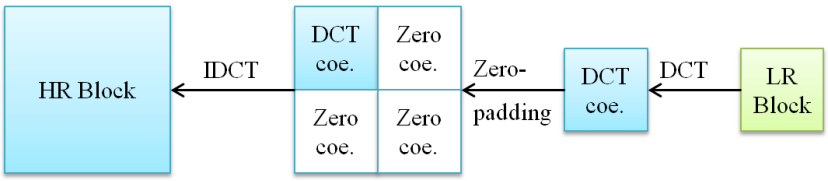


Fig. 2-6: Zero-padding.

Finally, the low frequency DCT coefficients $\bar{C}(k,l)$ with $0 \le k \le \frac{N}{2}-1, 0 \le l \le \frac{N}{2}-1$ are converted to spatial domain using the inverse DCT (IDCT).

$$x(m,n) = \frac{2}{M}\sum_{k=0}^{M-1}\sum_{l=0}^{M-1}\alpha(k)\alpha(l)(\bar{C}(k,l)\cos(\frac{(2m+1)\pi k}{2M})\cos(\frac{(2n+1)\pi l}{2M})), \qquad (2\text{-}34)$$

where $0 \le m,n \le M$ -1 (with $M = \frac{N}{2}$) and $\alpha(p) = \begin{cases} \sqrt{\frac{1}{2}} & \text{for } p = 0 \\ 1 & \text{otherwise} \end{cases}$.

2.4.2. ZERO-PADDING

Zero-padding is the simplest approach for DCT-based image up-sampling. A LR image patch is firstly converted to the DCT domain according to Eq. (2-32). As the high frequency part is missing, the high-frequency coefficients are all treated as zero coefficients. Then all the coefficients are approximated to the HR DCT coefficients.

$$\bar{C}(k,l) = 0.5 \times C(k,l), \qquad (2\text{-}35)$$

where $0 \le k,l \le N-1$.

Finally, the enlarged DCT block is converted back to spatial domain using IDCT.

2.4.3. HYBRID DCT-WIENER-BASED INTERPOLATION

Zero-padding can realize DCT block up-sampling, however, it is far from optimal to estimate the HR DCT block by simply considering the high frequency coefficients zeros. Unsurprisingly, the up-sampled image using zero-padding approach has strong ringing

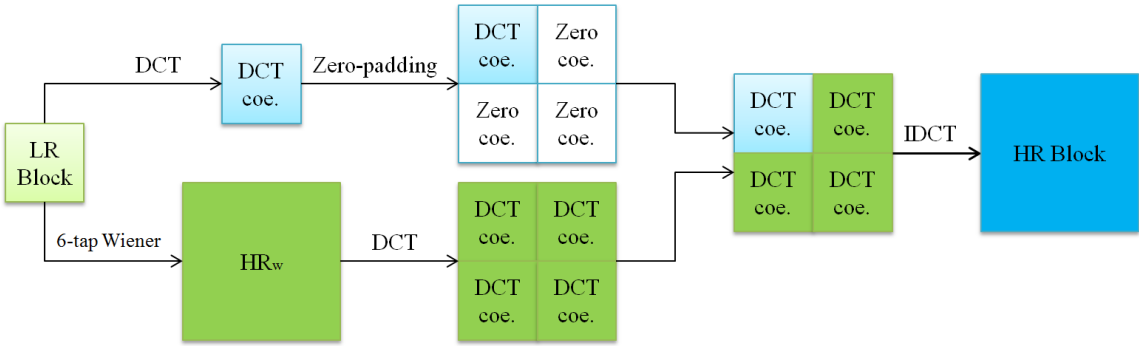


Fig. 2-7: Hybrid-Wiener-based image up-sampling approach.

artifacts near edges. A better solution is to estimate the high frequency coefficients, however, the difficulty is that coefficients in the DCT domain are highly uncorrelated which makes the inference from low frequency coefficients to high frequency coefficients a challenge problem. We must enhance the correlations from elsewhere, such as from spatial domain or by patch classification.

Wu, Yu and Chen [71] proposed a hybrid DCT-Wiener-Based image up-sampling algorithm which combines the original low frequency coefficients from the LR block and the high frequency coefficients spatially estimated using the 6-tap Wiener filter. As shown in Fig. 2-7, the upper part of the hybrid-wiener-based approach follows the zero-padding procedures which pad zeros on the high frequency positions and approximate the low frequency coefficients with a larger DCT block. In the lower part, the LR block is interpolated using the 6-tap Wiener filter. The coefficients of the 6-tap Wiener filter are shown below.

$$\frac{1}{32}[1, -5, 20, 20, -5, 1]. \tag{2-36}$$

Then the HR block spatially interpolated by the 6-tap Wiener filter is converted to DCT domain. Its high frequency coefficients are combined with the low frequency coefficients from the DCT block after zero-padding. The final HR block is then obtained

by performing IDCT to the combined DCT block.

The hybrid-wiener-based approach made a good attempt for DCT-based image up-sampling, however, the drawback is that the 6-tap Wiener filter is not a suitable candidate to spatially up-sample the LR block for DCT-based image up-sampling since there is a quarter-pixel shift between the 6-tap Wiener interpolated image and the original HR image. The up-sampled HR image still has strong artifacts.

Hung and Siu [72] made a step further and tried to learn a global universal Wiener filter or adaptively learn a specialized Wiener filter based on a *k*-nearest neighbor approach using MMSE estimator. Using the learnt Wiener filter [72] is able to improve the PSNR performance for DCT-Weiner-based interpolation, however, there still exists some room for improvement. In [72] only one learnt Wiener filter is applied for all different patches in despite of the fact that different patches need different filters. In [73] each block in the testing image will need to train one adaptive Wiener filter which apparently is very time-consuming and requires a large database to store two hundred thousands of training pairs.

CHAPTER 3 FAST IMAGE INTERPOLATION VIA RANDOM FORESTS

3.1. INTRODUCTION

The motivation of this chapter is to design a fast image interpolation algorithm which can achieve real-time performance as well as pleasing and natural image interpolation results. The main limitation in computational speed of the edge-directed methods [17, 42, 44, 45, 47-50, 52-57] is that these methods have to on-line learn the interpolation model from the local image patches, while in the sparse representation model (SRM)-based methods [58, 60], optimizing the sparse representation of each image patch is time-consuming. Besides, the nonlocal autoregressive modeling (NARM) algorithm [58] has to learn from the input image, which increases the algorithm complexity. Due to these facts, most of the image interpolation methods used in commercial area are still the polynomial-based algorithms (i.e. Bi-linear or Bi-cubic). In order to bypass the on-line learning procedure and explore a learning-based image interpolation approach different from the SRM-based path, we propose to use random forests to effectively learn a set of image interpolation models and simplify the model searching process by just comparing several pairs of pixels for each image patch.

Random forests [86] is an ensemble learning algorithm for classification and regression and has recently attracted much attention in computer vision [87-93]. Gall *et al*. [90] combined generalized Hough transform and random forests to build discriminative class-specific part appearance codebooks and achieved extraordinary object detection accuracy and near real-time performance. Lepetit and Fua [87] proposed to use randomized trees classification in place of the *K*-means plus nearest neighbor classifier

for keypoint recognition and yielded a powerful real-time matching method. Dollar and Zitnick [93] achieved state-of-the-art edge detection results by using a structured random forests whereas hundreds times faster than state-of-the-art edge detection methods. In [92], a real-time algorithm for facial feature detection is proposed by applying conditional regression forests to estimate facial feature points. Random forests also have been applied to image classification [88], categorization and segmentation [89]. From these previous works, random forests is proven to have high accuracy and very low computation cost.

The rest of this chapter is organized as follows: Section 3.2 gives a background analysis for image interpolation using classification and regression. Section 3.3 introduces the general image interpolation method using the random forests, and Section 3.4 describes the proposed two-stage framework. Section 3.5 presents the results of our extensive experimental work and Section 3.6 draws a conclusion.

3.2. BACKGROUND ANALYSIS

The natural image patches can be considered as a mixture of multi-class of image patches. We assume that the image patches in the same class form a linear subspace where the relationship between the HR image patches and the LR image patches can be modeled by a linear mapping function. Based on the above mentioned fact and assumption, the image interpolation problem can be solved by a combination of classification and regression process. During the training phase, the training LR-HR patch pairs can be classified into a number of classes according to the appearances of the LR patches. The relationship between LR and HR patches is modeled as a regression problem to learn a linear mapping function for each class. Several classification techniques, such as Support Vector Regression [5], *k*-nearest neighbor (*k*-NN) [51], *K*-means clustering [6, 22, 31],

have been applied for image up-sampling. However, in the previous works the classification process and the regression process are mostly separated and the classification is controlled by some manually selected parameters, for example the number of classes. Although the classification-based image interpolation algorithms can produce pleasing results, their speed is limited by the classification methods. The SVR is computationally complex during both training and testing since the optimal hyper-parameters are explored among the whole training set. The *k*-NN method has to find *k* nearest neighbors in the training set and can be very time-consuming. Although the *K*-means clustering has classified the training data into *K* predefined clusters during training according to the Euclidean distance, the input image patch has to find the closest cluster center during runtime and the matching process is not efficient.

In this chapter, random forests is applied for fast image interpolation by recursively classifying the LR patches into one of the leaf nodes using a simple decision (comparing two pixel values with a threshold) in each non-leaf node and mapping LR patches to HR patches through the linear regression models stored in the leaf nodes. Besides, different from the *K*-means clustering algorithm which performs classification by minimizing l_2 distance, the proposed approach achieves classification via minimizing the reconstruction MSE which relates directly to Peak Signal-to-Noise Ratio ($\mathrm{PSNR} = 20\log_{10}\frac{255}{\sqrt{MSE}}$).

3.3. RANDOM FORESTS FOR IMAGE INTERPOLATION

Random forests [86] $\mathsf{T} = \{T_t\}$ consists of a set of decision trees which can jointly

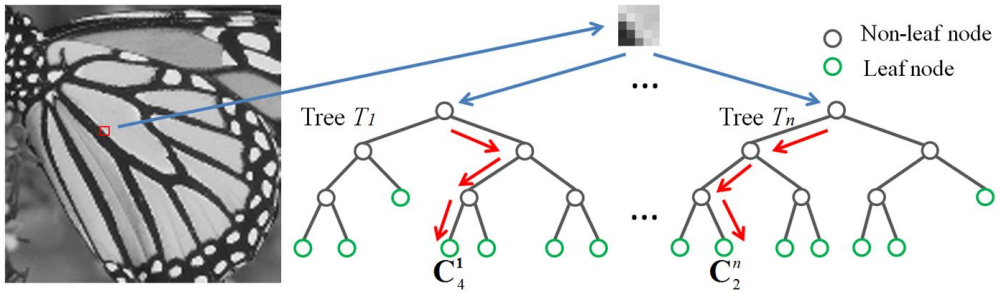


Fig. 3-1: A random forests which consists of a set of decision trees. Each decision tree recursively classifies the input LR patch into left or right child node, until a leaf node is reached. By using the linear regression models (e.g. $C_4^1,\ldots,C_2^n$) stored in the reach leaf node, the input LR patch can be projected to the HR patch space.

realize classification and regression. Each decision tree contains non-leaf nodes and leaf nodes as shown in Fig. 3-1. The non-leaf nodes evaluate the LR patch arrived at this node by its appearances according to the stored binary test and pass this LR patch to its left or right child node until a leaf node is reached. Each leaf node contains a linear regression model, and the corresponding HR patch of the input LR patch can be estimated by the regression model. As discussed in Section 3.2, the image interpolation can be viewed as a classification and regression process. For classification, the binary test on each non-leaf node recursively classifies the LR patch into one of the leaf nodes. For regression, the reached leaf node in each individual tree has a regressor to map the input LR patch to HR patch.

3.3.1. TRAINING SCHEME

In the training phase, each non-leaf node in the decision tree chooses one binary test from a set of randomly selected binary tests, which can maximize the error reduction, to split the training data at this node into its two child nodes. When the error reduction is less than zero, that node is declared as a leaf node. Hence a regression model is learned using the LR-HR patch pairs which reached this leaf node.

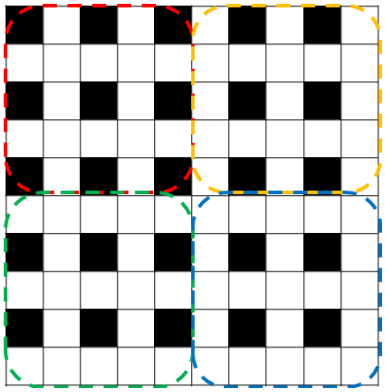


Fig. 3-2: The black points are the pixels remained after the direct down-sampling from the original high-resolution image, and the white points are the unknown pixels to be interpolated. There are four different patterns in the interpolated image with respect to the structure of the known pixels for two times interpolation.

3.3.1.1. TRAINING DATA GENERATION

The training time of a decision tree is exponentially related to the training data size. To relieve the burden on training, we propose two strategies for training data generation.

First, the training data is divided into different groups as illustrated in Fig. 3-2. In the degradation process of image interpolation problem, $1/p^2$ of the pixels in the resultant image comes from the original HR image for an image interpolation of p times. When the image is interpolated by a factor of 2, there will be four different patterns of the known pixels (marked in red, yellow, green and blue rectangles in Fig.3-2). The regression models for these four patterns should be different from each other, since the positions of known pixels are different. Based on this feature of image interpolation, the training data are divided into four groups according to the known pixel pattern and the training time can be reduced by 9 times.

Second, the focus of training is on the image patches where edge pixels can be detected. During training and interpolation, the LR image is initially interpolated to the same size

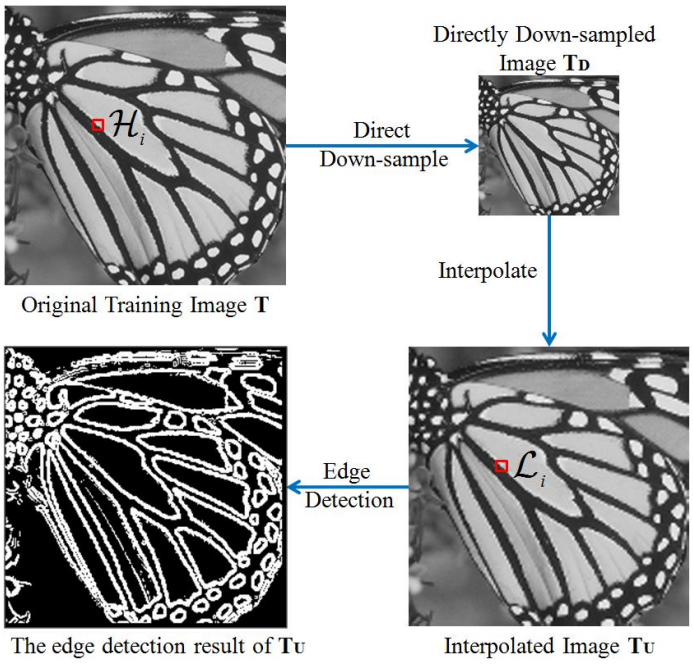


Fig. 3-3: Training data generation process.

as the desired HR image using Bi-cubic interpolation. The Bi-cubic interpolation will introduce artifacts along strong edges, while performs well in smooth regions. Hence, we only collect patches containing edge pixels, while exclude patches from smooth regions in order to reduce both the training time and testing time. The edge pixels are determined by applying the Canny edge detection on the Bi-cubic interpolated image. The objective of applying the Canny edge detection on the Bi-cubic interpolated image is not to accurately detect the edges, but to detect the possible regions where there are artifacts introduced by Bi-cubic interpolation. If the edge magnitude of a pixel is larger than a certain threshold, this pixel is regarded as an edge pixel. The edge threshold is determined by considering both the testing time and the image interpolation quality. With an increase in edge threshold, the image interpolation quality would decrease, since fewer prominent edges would be processed. We select 60 as the threshold, because there is no significant drop in PSNR compared with processing the whole image, while it reduces to half of its

processing time.

The extracted LR-HR patch pairs $\{P_i = (\mathbf{L}_i, \mathbf{H}_i)\}$ are used for training, where $\mathbf{L}_i \in \mathbf{R}^d$ is the corresponding vectorized LR patch on $\mathbf{T_U}$ which is interpolated from the $\mathbf{T}$ directly down-sampled image $\mathbf{T_D}$ (using Bi-cubic interpolation or Stage 1 interpolation framework of the proposed work), $\mathbf{H}_i \in \mathbf{R}^d$ is the vectorized HR patch sampled from the original training image $\mathbf{T}$, and $\sqrt{d}$ is the patch size, as shown in Fig. 3-3.

3.3.1.2. LINEAR REGRESSION MODEL

At each node, there is a linear regression model to fit the relationship between HR and LR patches for the training data reached this node. Assume l training patch pairs $S_j = \{P_i \mid i = 1,...,l\}$ arrived at node j. We group all the training patches into matrix forms $\mathbf{L} \in \mathbf{R}^d \times \mathbf{R}^l$ and $\mathbf{H} \in \mathbf{R}^d \times \mathbf{R}^l$ for LR and HR training patches, respectively. The linear regression model $\mathbf{C}_j \in \mathbf{R}^d \times \mathbf{R}^d$ at node j can be obtained by minimizing the MSE between the original HR patches $\mathbf{H}$ and the reconstructed HR patches $\mathbf{H}^R$:

$$\mathbf{H}^R = \mathbf{C}_j \mathbf{L}. \tag{3-1}$$

$$\mathbf{C}_j = \arg\min_{\mathbf{C}_j} \| \mathbf{H} - \mathbf{C}_j \mathbf{L} \|_2^2 . \tag{3-2}$$

Eq. (3-2) has a closed form solution and can be solved by the least squares method.

$$\mathbf{C}_j = \mathbf{H}\mathbf{L}^{\mathrm{T}} (\mathbf{L}\mathbf{L}^{\mathrm{T}})^{-1}. \tag{3-3}$$

3.3.1.3. BINARY TEST AND SPLIT FUNCTION

Every non-leaf node in the decision tree is associated with a binary test to split the training data into its left or right child node. In this chapter, the binary test $\theta = (\phi, \boldsymbol{\psi}, \tau)$ where $\phi = \phi(\mathbf{L})$ selects two pixel in the LR patch vector $\mathbf{L} \in \mathbf{R}^d$ at position p and q, and

$\boldsymbol{\psi} = (1,-1)$. This kind of binary tests has the intensity invariant property which is helpful to adapt to various intensity conditions in natural images and increase training efficiency.

The split function $h(\mathbf{L}, \theta)$ splits the LR-HR patch pair into left or right node according to the appearance of the LR patch $\mathbf{L} \in \mathbf{R}^d$ together with the randomly generated binary test parameter θ as:

$$h(\mathbf{L}, \theta) = \begin{cases} 0, & \text{if } \mathbf{L}(p) < \mathbf{L}(q) + \tau \\ 1, & \text{otherwise} \end{cases}, \tag{3-4}$$

where $1 \le p, q \le d$, $0 \le \tau \le 255$ and d is the vector length of $\mathbf{L}$.

The extremely randomized trees [85] suggested that, for regression problems, the number of randomly selected binary tests K should be equal to the number of attributes of the observed feature vector. Since in this chapter each binary test samples two pixel positions in the LR patch, the number of randomly selected binary tests at each node should be:

$$K = (d-1)d, \tag{3-5}$$

where d is the vector length of the LR patch.

3.3.1.4. DECISION TREE GROWING

At node j, each binary test θ_i of the K randomly selected binary tests $\Theta = \{\theta_i \mid i = 1,..., K\}$ divides the training data $\mathbf{S}_j$ into its left child node $\mathbf{S}_j^L$ and right child node $\mathbf{S}_j^R$, exclusively.

$$\begin{aligned} \mathbf{S}_j^L &= \{P_k \mid h(\mathbf{L}_k, \theta_i) = 0, \mathbf{P}_k \in \mathbf{S}_j\} \\ \mathbf{S}_j^R &= \mathbf{S}_j \setminus \mathbf{S}_j^L \end{aligned}. \tag{3-6}$$

The number of training patch pairs at each child node should be larger than a predefined minimum number of training data N_{min} (usually set to 5 to 10 times as the

number of the minimum training data required to train a MMSE estimator) to effectively learn a regression model and avoid over-fitting. If a split cannot satisfy Eq. (3-7), that split should be rejected (let us define $|\mathbf{S}|$ as the number of training data at $\mathbf{S}$).

$$\min(|\mathbf{S}_j^L|, |\mathbf{S}_j^R|) \geq N_{min}. \tag{3-7}$$

The regression models for training data $\mathbf{S}_j$, $\mathbf{S}_j^L$ and $\mathbf{S}_j^R$ are learned according to Eq. (3-3). Using the learned regression models, the reconstructed HR patches can be obtained through Eq. (3-1). Let us define the fitting error $E(\mathbf{S})$ as the mean squared error between the original HR patches **H** and the reconstructed HR patches $\mathbf{H}^R$ of training data $\mathbf{S}$.

$$E(\mathbf{S}) = \frac{1}{|\mathbf{S}|} \| \mathbf{H} - \mathbf{H}^R \|_2^2 . \tag{3-8}$$

The goodness of a binary test θ_i applied on training data $\mathbf{S}_j$ is evaluated through the error reduction $R(\mathbf{S}_j, \theta_i)$.

$$R(\mathbf{S}_j, \theta_i) = E(\mathbf{S}_j) - \sum_{n \in \{L.R\}} \frac{|\mathbf{S}_j^n|}{|\mathbf{S}_j|} E(\mathbf{S}_j^n). \tag{3-9}$$

The best binary test θ_j is the one that achieves the maximum error reduction among all *K* selected binary tests Θ.

$$\theta_j = \arg \max_{\theta \in \Theta} R(\mathbf{S}_j, \theta). \tag{3-10}$$

If $R(\mathbf{S}_j, \theta_i)$ is larger than zero, node *j* will be regarded as a non-leaf node and the decision tree will keep growing through this node; otherwise, this node is declared as a leaf node. Each non-leaf node stores its best binary test parameter and each leaf node stores the learned linear regression model. A decision tree is gradually constructed by recursively partitioning the training data at each node into its two child nodes using the

obtained best binary test.

3.3.1.5. CONSTRAINT ON SPLITS

We propose to apply a constraint to control the relative training data size of two child nodes. A randomly selected threshold τ in Eq. (3-4) would divide the training data at parent node into two unbalanced child nodes (two child nodes have large difference in training data size). Random forests tries to maximize the error reduction which may result in a low fitting error at the large child node and a high fitting error at the small child node. The little drop of the fitting error at the large child node will mask the increase of fitting error at the small child node and result in a high error reduction. However, a child node with large fitting error will affect the prediction accuracy of the decision tree, since a large fitting error means the patches at that node have low similarity. Furthermore, the fitting error of its child node may even be larger. To increase training effectiveness, we propose a constraint to restrict the relative size of two child nodes; only the split which satisfies the constraint is used.

$$\max(|S_j^L|,|S_j^R|)\times\lambda \le \min(|S_j^L|,|S_j^R|). \tag{3-11}$$

where $\lambda \in [0,1]$ controls the proportion between the training data sizes of two child nodes.

TABLE 3-1

ALGORITHM 1 IMAGE INTERPOLATION VIA RANDOM FORESTS TRAINING SCHEME

1. Direct down-sample a subset of the training images **T** by a factor of 2, and obtain the down-sampled images $\mathbf{T_D}$.
2. Interpolate $\mathbf{T_D}$ by a factor of 2 (using bi-cubic interpolation or the proposed Stage 1 interpolation framework), and obtain the interpolated images $\mathbf{T_U}$.
3. Extract training patch LR-HR pairs from $\mathbf{T_U}$ and **T** on the edge areas and classify into 4 groups with respect to the known pixel patterns.
4. **For** each group of training patch pairs:
   5. Initiate the root node with all extracted training data S_0.
   6. **For** each unprocessed non-leaf node *j* and the training data S_j:
      7. **If** the training data size $|S_j| < 2 \times Nmin$
         8. Declare this node as a leaf node, store the regression model learned using S_j and go to step 6;
      Else if
         9. Randomly select *K* binary tests $\Theta = \{\theta_i \mid i = 1,..., K\}$, which all satisfy Eq. (3-7) and Eq. (3-11).
         10. Find the best binary test θ_j according to Eq. (3-8), (3-9) and (3-10).
         11. **If** $R(S_j, \theta_j) > 0$
            12. Declare this node as a non-leaf node, store the best binary test θ_j and split the training data according to θ_j
         Else if
            13. otherwise declare this node as a leaf node and store the regression model learned using S_j.
         End if
      End if
   End for
End for

Actually, λ controls the growing style of the decision trees. The smaller λ is, the more unbalanced decision trees tend to be obtained, and vice versa. By properly limiting the relative size between two child nodes, we can gradually divide the whole training data into a number of leaf nodes with small MSE, thus the interpolation results will achieve higher PSNR and the tree structure is more compact. The experimental results for choosing the optimal λ values will be presented in Section 3.5.1.

The pseudo-code description of the proposed image interpolation via random forests

TABLE 3-2

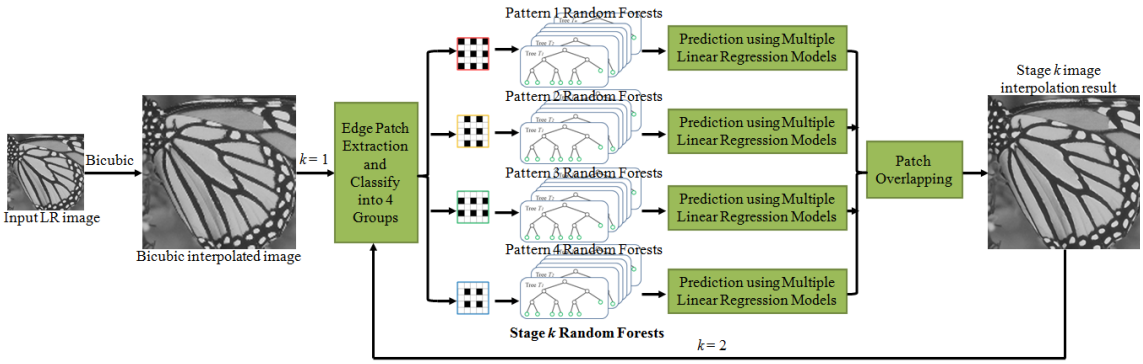
ALGORITHM 2 IMAGE INTERPOLATION VIA RANDOM FORESTS INTERPOLATION SCHEME

1. Interpolate the input LR image $\mathbf{T_D}$ by a factor of 2 (using bicubic interpolation or the proposed Stage 1 interpolation framework), and obtain the initial interpolated image $\mathbf{T_U}$.
2. Extract patches from $\mathbf{T_U}$ on the edge areas and classify the patches into 4 groups with respect to the known pixel patterns.
3. **For** each extracted patch $\mathbf{L}$ in group i:
   4. **For** each decision tree T_t^i in the group i Random Forests:
      5. Classify patch $\mathbf{L}$ into left or right child node according to the binary test stored at each non-leaf node, until a leaf node is reached.
      6. Store the regression model at the reached leaf node.
   End for
   7. Get the reconstructed HR image patch $\mathbf{H}^R$ using Eq. (3-12).
End for
8. Get the final interpolated image by overlapping reconstructed HR image patches.

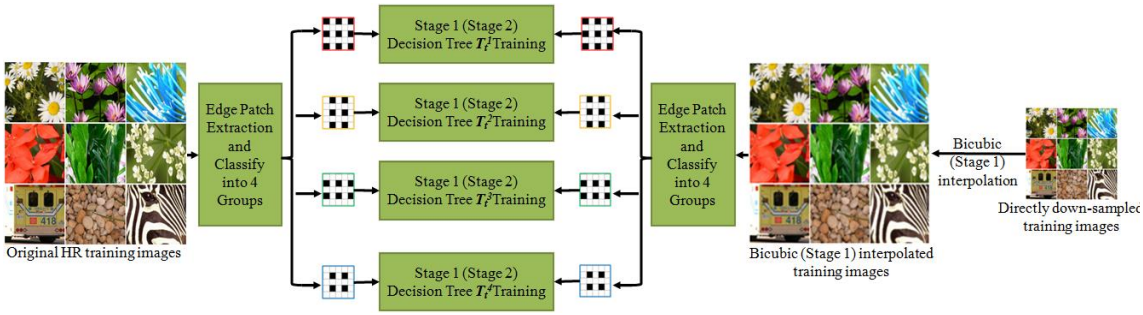
training scheme is given in TABLE 3-1 **Algorithm 1**.

3.3.2. INTERPOLATION SCHEME

In the image interpolation phase, the input LR image is initially interpolated to the same size as the desired HR image (using Bi-cubic interpolation or Stage 1 up-sampling framework of the proposed approach). Image patches containing edge pixels are extracted from the initial interpolated image, and divided into four groups according to the known pixel patterns. Each group of image patches are passed into the random forests learned using the training data with the same known pixel patterns. In each decision tree, the input patch $\mathbf{L}$ is classified into left or right node according to the binary test stored at each non-leaf node. When the patch reaches a leaf node, the linear regression model $\mathbf{C}$ stored at the leaf node is used for the interpolation. In order to efficiently combine all the reconstruction results generated from multiple decision trees, the obtained N linear regression models are firstly averaged. The final reconstructed HR patch is obtained by multiplying the averaged linear regression model with the patch $\mathbf{L}$.



(a) Two-stage interpolation scheme



(b) Two-stage training scheme

Fig. 3-4: Flowcharts of the proposed two-stage image interpolation framework: (a) interpolation scheme and (b) training scheme.

$$\mathbf{H}^R = (\frac{1}{N}\sum_{i=1}^{N}\mathbf{C}_i)\mathbf{L}. \tag{3-12}$$

As the input patches are overlapped with their neighbors, each pixel will have multiple prediction values. The final interpolated HR image is obtained by overlapping the reconstructed HR patches (averaging all prediction values of a pixel) to preserve the consistence between neighboring pixels.

The proposed image interpolation via random forests up-sampling scheme is summarized in TABLE 3-2 **Algorithm 2**.

3.4. TWO-STAGE IMAGE INTERPOLATION FRAMEWORK

The decision tree needs a huge amount of training data. As the tree depth increases by one, the required training data doubles. With a limited number of training data, we propose a novel two-stage image interpolation framework to further explore the correlation

between the error residues and the original HR images after the random forests learning. The two-stage image interpolation framework breaks the performance limit of the original random forests and brings about 0.2 dB improvement in PSNR.

Fig. 3-4 shows our proposed two-stage image interpolation framework. In the interpolation scheme Fig.3-4(a), the input LR image is initially interpolated by Bi-cubic interpolation. The patches with edge pixels in the Bi-cubic interpolation image will be extracted, classified into 4 groups according to the known pixel pattern and passed into the Stage 1 random forests for classification. Then the HR patch of each input patch is predicted with regression models from Stage 1 random forests. The Stage 1 interpolated image is obtained by overlapping reconstructed HR patches. (The above procedures follow the description in **Algorithm 2**.) Using the Stage 2 random forests and taking the Stage 1 image interpolation result as the input, the Stage 2 image interpolation result can be computed similarly. The Stage 1 image interpolation result is better than the results of the state-of-the-art image interpolation methods, while the Stage 2 image interpolation scheme further refines Stage 1 interpolated image.

The training scheme of the proposed two-stage framework is shown in Fig. 3-4 (b). For each decision tree, only a subset of the whole training images will be used for training. The Stage 1 decision tree T_t^i uses the patch pairs which contain edge pixels of known pixel pattern i from the original HR image and its direct down-sampled and then Bi-cubic interpolated image for training. (The above procedures follow the description in **Algorithm 1**.) While the training data of Stage 2 decision tree T_t^i are the i^{th} group known pixel pattern patch pairs extracted from the edge areas in the original HR images and the Stage 1 interpolated image. Stage 1 random forests learns classification for the strong

Fig. 3-5: 18 test images. From left to right and top to bottom: *Bears*, *Bicycle*, *Boat*, *Butterfly*, *Cameraman*, *Coala*, *Elk*, *Fence*, *Flowers*, *Foreman*, *Girl*, *House*, *Leaves*, *Lena*, *Parrot*, *Parthenon*, *Starfish* and *Stream*.

Fig. 3-6: Example training images.

ringing and aliasing patches in the Bi-cubic interpolated image and regression models to map them to HR patches. Stage 2 random forests learns the relationship between the patches in the original HR image and the Stage 1 interpolated image to refine small artifacts remained in the results of Stage 1 image interpolation.

3.5. EXPERIMENTAL RESULTS

In this section we evaluate the proposed fast image interpolation via random forests (FIRF) method using 18 commonly used test images *Bears*, *Bicycle*, *Boat*, *Butterfly*, *Cameraman*, *Coala*, *Elk*, *Fence*, *Flowers*, *Foreman*, *Girl*, *House*, *Leaves*, *Lena*, *Parrot*,

Parthenon, *Starfish* and *Stream*. As shown in Fig. 3-5, these 18 test images contain various edges and texture contents. All LR images were obtained by directly down-sampling the original HR images by a factor of 2, as in image interpolation papers [44, 48, 50, 51, 55, 57, 58, 60]. The PSNR along with two perceptual quality metrics, SSIM (Structural SIMilarity) [94] and FSIM (Feature SIMilarity) [95], are used to jointly evaluate the performance of the proposed FIRF method and other methods.

In our implementation, image patch size $\sqrt{d}$ was selected as 5 and the minimum number of training data, N_{min} , is 200. During the training, for each decision tree in the random forests, we sampled around 650000 LR-HR patch pairs from natural images collected from the internet. Parts of the training images are shown in Fig. 3-6. After training, each decision tree will have about 2400 leaf nodes.

To comprehensively evaluate the proposed FIRF method, we initially give the experimental results for choosing the optimal λ value for the proposed constraint. Then the influence of the decision trees number in the random forests and the effectiveness of the proposed two-stage image interpolation scheme will be presented. Finally, the proposed FIRF method will be compared with the state-of-the-art methods. For easy notation, the proposed FIRF method using n decision trees per stage and k stage image interpolation scheme is denoted as FIRF(n, k).

3.5.1. CONSTRAINT ON SPLITS

In Section 3.3.1.5, we proposed to control the relative size of two child nodes by varying the λ value, as stated in Eq. (3-11). The PSNR (dB) of FIRF(1,1) image interpolation results with respect to different λ values are shown in Table 3-3. We have found that when $\lambda = 0$ (no constraint is applied) the average PSNR is the lowest. By

TABLE 3-3
PSNR (DB) OF THE PROPOSED FIRF(3,1) METHOD WITH DIFFERENT λ

Testing Images	**λ=0**	**λ=0.2**	**λ=0.4**	**λ=0.6**	**λ=0.8**
Bears	28.69	28.72	28.70	28.70	**28.71**
Bicycle	21.36	21.45	21.46	21.48	21.48
Boat	27.89	27.96	27.96	28.01	28.03
Butterfly	30.17	30.29	30.36	30.35	30.35
Cameraman	26.32	26.35	26.37	26.41	26.36
Coala	33.79	33.89	33.88	33.89	33.89
Elk	33.28	33.49	33.52	33.54	33.54
Fence	24.47	24.51	24.49	24.49	24.50
Flowers	28.85	28.97	28.97	28.98	28.97
Foreman	37.75	38.07	38.16	38.15	38.12
Girl	34.01	34.07	34.08	34.08	34.09
House	33.29	33.50	33.57	33.58	33.57
Leaves	29.25	29.35	29.40	29.46	29.46
Lena	34.56	34.65	34.64	34.65	34.65
Parrot	27.13	27.12	27.18	27.19	27.18
Parthenon	27.41	27.44	27.46	27.44	27.45
Starfish	31.27	31.43	31.45	31.45	31.48
Stream	26.07	26.07	26.08	26.07	26.06
Average	29.75	29.85	29.87	29.88	29.88

increasing λ, the average PSNR grows and tends to be saturated after $\lambda > 0.4$ achieving 0.13 dB improvement better than that for $\lambda = 0$. However, a large λ value gives a tight constraint which requires longer training time. To balance the training effectiveness and the training time, we chose $\lambda = 0.7$ in our experimental works as follows.

3.5.2. INFLUENCE OF THE DECISION TREE NUMBER AND EFFECTIVENESS OF THE TWO-STAGE IMAGE INTERPOLATION FRAMEWORK

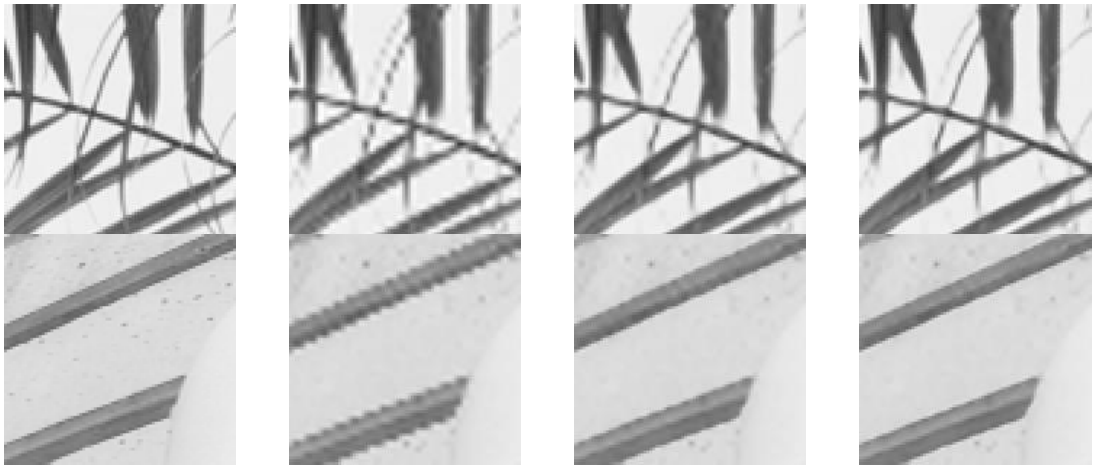
In this part, the proposed FIRF method are evaluated in terms of various decision tree numbers with Stage 1 and Stage 2 image interpolation schemes. Table 3-4 report the PSNR (dB) and average computational time for the proposed FIRF(*n*, *k*) method, where *n* varies from 1 to 5 and *k* equals to 1 or 2.

When the number of decision trees *n* increases from 1 to 3, the average PSNR of the test images improves fast for both Stage 1 and Stage 2 image interpolation schemes. This reflects the unstable characteristics of the single decision tree. The image interpolation

TABLE 3-4
PSNR (dB) AND AVERAGE COMPUTATIONAL TIME FOR PROPOSED FIRF(n, k) METHOD WHERE n=1,…, 5; k=1, 2.

Testing Images	FIRF(1,1)	FIRF(2,1)	FIRF(3,1)	FIRF(4,1)	FIRF(5,1)	FIRF(1,2)	FIRF(2,2)	FIRF(3,2)	FIRF(4,2)	FIRF(5,2)
Bears	28.72	28.75	28.75	28.75	28.76	28.70	28.76	28.77	28.78	28.79
Bicycle	21.51	21.50	21.48	21.48	21.48	21.76	21.81	21.79	21.77	21.78
Boat	28.01	28.10	28.11	28.12	28.12	28.06	28.15	28.22	28.27	28.27
Butterfly	30.31	30.37	30.41	30.43	30.43	30.44	30.55	30.65	30.68	30.68
Cameraman	26.36	26.35	26.35	26.37	26.38	26.55	26.57	26.57	26.56	26.57
Coala	33.88	33.88	33.91	33.92	33.93	33.96	33.99	34.02	34.03	34.02
Elk	33.52	33.47	33.47	33.48	33.51	33.95	33.90	33.88	33.87	33.88
Fence	24.48	24.58	24.65	24.64	24.64	24.39	24.48	24.60	24.61	24.61
Flowers	28.98	28.99	29.00	29.00	29.01	29.13	29.16	29.17	29.17	29.18
Foreman	38.13	38.15	38.16	38.18	38.17	38.41	38.50	38.53	38.54	38.54
Girl	34.07	34.10	34.10	34.10	34.10	34.07	34.11	34.11	34.12	34.12
House	33.56	33.62	33.62	33.61	33.60	33.93	34.05	34.06	34.04	34.05
Leaves	29.48	29.61	29.64	29.65	29.65	29.60	29.84	29.96	30.00	30.00
Lena	34.64	34.68	34.72	34.73	34.73	34.61	34.69	34.76	34.79	34.79
Parrot	27.21	27.24	27.27	27.28	27.29	27.23	27.29	27.36	27.37	27.38
Parthenon	27.44	27.48	27.50	27.50	27.50	27.45	27.52	27.56	27.57	27.57
Starfish	31.45	31.48	31.50	31.52	31.53	31.68	31.78	31.83	31.85	31.87
Stream	26.07	26.10	26.10	26.11	26.11	26.04	26.09	26.11	26.12	26.12
Average PSNR	**29.88**	**29.91**	**29.93**	**29.94**	**29.94**	**30.00**	**30.07**	**30.11**	**30.12**	**30.12**
Time (ms)	**87.50**	**138.61**	**192.28**	**259.11**	**304.39**	**173.28**	**278.11**	**384.78**	**489.56**	**600.56**

results become more and more stable by combining the regression models from multiple de-correlated decision trees. However, higher computational load is required with the growing number of decision trees. The average PSNR for both Stage 1 and Stage 2 image interpolation results tend to saturate when the decision tree number exceeds 3. The saturated PSNR value for Stage 2 image interpolation results is about 0.2 dB higher than that of Stage 1. The overall two-stage image interpolation scheme requires about twice the computational load of the Stage 1 image interpolation scheme, because the total number of decision trees used is doubled. Thus, the proposed FIRF(n,k) can provide scalable image interpolation by varying n and k according to the computational power of the devices. The subjective comparison for Stage 1 and Stage 2 image interpolation results are shown in Fig. 3-7. We can see that Stage 1 image interpolation scheme can remove most of the artifacts in the Bi-cubic interpolated images. With one more stage for



(a) Original HR image (b) Bicubic (c) Stage 1 results (d) Stage 2 results

Fig. 3-7: Reconstructed HR images by Stage 1 and Stage 2 image interpolation scheme. (a) Original HR image. (b) Bicubic. (c) Proposed FIRF(3,1). (d) Proposed FIRF(3,2).

refinement, the Stage 2 produces more natural images with sharper and higher continuity edges. From both objective and subjective comparison, the effect of the proposed two-stage image interpolation framework is demonstrated.

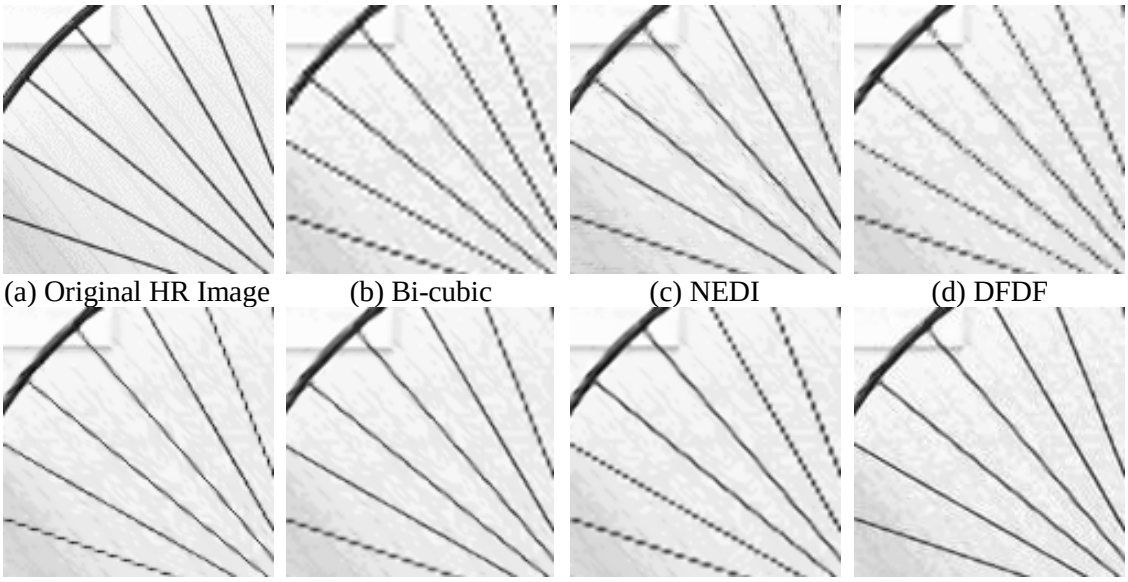
3.5.3. COMPARISON WITH STATE-OF-THE-ARTS

Several state-of-the-art image interpolation methods are selected for comparison, including the Bi-cubic, new edge-directed interpolation (NEDI) method [44], directional filtering and data fusion (DFDF) method [48], regularized local linear regression (RLLR) method [55], robust soft-decision (RSAI) method [57], nonlocal autoregressive modeling (NARM) method [58], bilateral soft-decision (BSAI) method [56] and iterative curve based interpolation (ICBI) method [17]. Among all these approaches, the NEDI, DFDF, RLLR and RSAI are outstanding edge directed image interpolation methods developed in recent years. The NARM is a sparse representation model based image interpolation method proposed recently. The BSAI and ICBI are two well-known fast image interpolation methods.

TABLE 3-5

PSNR (dB), SSIM AND FSIM RESULTS BY DIFFERENT INTERPOLATION METHODS

Testing Images	**Bicubic**	**NEDI [20]**	**DFDF [24]**	**RLLR [28]**	**RSAI [27]**	**NARM [33]**	**BSAI [26]**	**ICBI [29]**	**FIRF(1,1) Proposed**	**FIRF(3,2) Proposed**
Bears	28.47	28.08	28.18	28.62	28.67	28.30	28.45	27.68	28.72	**28.77**
	0.8796	0.8688	0.8687	0.8771	0.8818	0.8799	0.8786	0.8718	0.8815	**0.8829**
	0.9339	0.9253	0.9276	0.9302	**0.9377**	0.9341	0.9321	0.9320	0.9339	0.9341
Bicycle	20.27	20.81	20.39	20.70	21.17	20.46	20.97	19.55	21.51	**21.79**
	0.7559	0.7761	0.7606	0.7877	0.7904	0.7837	0.7899	0.7380	0.7934	**0.8022**
	0.9236	0.9260	0.9233	0.9286	0.9313	0.9282	0.9296	0.9108	0.9339	**0.9365**
Boat	27.22	27.38	27.43	27.69	27.90	27.88	27.87	26.81	28.01	**28.22**
	0.8247	0.8293	0.8276	0.8416	0.8421	**0.8477**	0.8435	0.8164	0.8381	0.8425
	0.9317	0.9310	0.9332	0.9372	0.9394	**0.9416**	0.9388	0.9306	0.9399	0.9408
Butterfly	27.70	27.35	28.66	29.29	29.22	30.35	29.36	28.59	30.31	**30.65**
	0.9241	0.9321	0.9397	0.9480	0.9472	0.9559	0.9488	0.9366	0.9536	**0.9565**
	0.9156	0.9282	0.9451	0.9486	0.9473	0.9571	0.9541	0.9311	0.9599	**0.9630**
Cameraman	25.37	25.42	25.67	25.75	26.00	25.92	25.96	25.16	26.36	**26.57**
	0.8627	0.8626	0.8670	0.8714	0.8730	0.8760	0.8750	0.8611	0.8764	**0.8795**
	0.9036	0.9059	0.9143	0.9156	0.9163	0.9201	0.9195	0.9099	0.9230	**0.9262**
Coala	33.22	32.77	33.00	33.62	33.83	33.83	33.62	33.13	33.88	**34.02**
	0.9137	0.9030	0.9040	0.9106	0.9155	0.9138	0.9119	0.9117	0.9168	**0.9177**
	0.9556	0.9492	0.9520	0.9542	0.9577	0.9567	0.9555	0.9573	0.9589	**0.9593**
Elk	31.59	32.37	31.75	32.87	33.24	33.25	32.86	31.54	33.52	**33.88**
	0.9249	0.9291	0.9230	0.9345	0.9368	0.9356	0.9335	0.9230	0.9366	**0.9387**
	0.9480	0.9530	0.9502	0.9569	0.9581	0.9575	0.9558	0.9493	0.9592	**0.9606**
Fence	24.54	22.94	24.55	24.00	24.44	**24.70**	24.51	23.75	24.48	24.60
	0.7783	0.7584	0.7758	0.7755	0.7833	**0.7924**	0.7830	0.7612	0.7814	0.7855
	0.8828	0.8823	0.8791	0.8778	0.8803	**0.9037**	0.8773	0.9017	0.8759	0.8805
Flowers	28.09	28.25	28.40	28.89	28.94	28.78	28.88	27.96	28.98	**29.17**
	0.8150	0.8277	0.8223	0.8369	0.8368	0.8360	0.8351	0.8126	0.8326	**0.8371**
	0.9198	0.9227	0.9241	0.9288	0.9297	0.9293	0.9291	0.9201	0.9301	**0.9319**
Foreman	35.56	35.90	36.81	37.70	37.67	**38.60**	37.53	36.20	38.13	38.53
	0.9488	0.9532	0.9541	0.9570	0.9577	0.9574	0.9564	0.9493	0.9571	**0.9582**
	0.9652	0.9700	0.9712	0.9740	0.9745	0.9754	0.9727	0.9675	0.9750	**0.9764**
Girl	33.84	33.84	33.80	34.33	**34.34**	34.28	34.31	33.59	34.07	34.11
	0.8534	0.8572	0.8523	0.8637	**0.8633**	0.8612	0.8630	0.8467	0.8563	0.8569
	0.9416	0.9410	0.9395	0.9441	**0.9450**	0.9435	0.9437	0.9395	0.9434	0.9434
House	32.17	31.67	32.57	32.93	32.92	33.45	32.95	31.82	33.56	**34.06**
	0.8768	0.8743	0.8775	0.8836	0.8853	0.8834	0.8862	0.8661	0.8855	**0.8882**
	0.9404	0.9434	0.9478	0.9498	0.9505	0.9551	0.9509	0.9401	0.9558	**0.9604**
Leaves	26.88	26.23	27.22	28.43	28.66	29.77	28.60	27.37	29.48	**29.96**
	0.9363	0.9403	0.9433	0.9562	0.9574	0.9670	0.9587	0.9436	0.9639	**0.9678**
	0.9262	0.9429	0.9478	0.9605	0.9590	0.9670	0.9613	0.9388	0.9655	**0.9683**
Lena	33.92	33.76	33.89	34.47	34.73	**35.04**	34.61	33.97	34.64	34.76
	0.914	0.9134	0.9122	0.9190	0.9201	**0.9238**	0.9208	0.9111	0.9172	0.9180
	0.9870	0.9868	0.9867	0.9883	0.9887	**0.9892**	0.9886	0.9860	0.9881	0.9882
Parrot	26.22	26.10	26.38	26.83	26.73	26.90	26.81	26.31	27.21	**27.36**
	0.8852	0.8858	0.8859	0.8952	0.8953	0.8936	0.8933	0.8850	0.8956	**0.8970**
	0.9290	0.9320	0.9334	0.9408	0.9398	0.9413	0.9400	0.9346	0.9425	**0.9434**
Parthenon	27.08	26.79	27.18	27.14	27.22	27.26	27.34	26.42	27.44	**27.56**
	0.8037	0.7880	0.8032	0.8013	0.8052	0.8059	0.8114	0.7797	0.8110	**0.8138**
	0.8950	0.8910	0.8963	0.8960	0.8978	0.9021	0.8971	0.8930	0.9016	**0.9030**
Starfish	30.25	29.36	30.07	30.41	30.78	31.73	31.22	30.19	31.45	**31.83**
	0.9168	0.8985	0.9118	0.9140	0.9203	0.9302	0.9258	0.9167	0.9287	**0.9324**
	0.9521	0.9458	0.9543	0.9561	0.9581	0.9646	0.9615	0.9548	0.9633	**0.9657**
Stream	25.83	25.60	25.66	25.98	26.03	25.84	25.90	25.32	26.07	**26.11**
	0.7974	0.7812	0.7866	0.7948	0.7999	0.8008	0.7966	0.7871	0.8007	**0.8018**
	0.9580	0.9542	0.9561	0.9584	0.9591	0.9591	0.9591	0.9541	0.9598	**0.9600**
Average	28.79	28.59	28.98	29.43	29.58	29.80	29.54	28.63	29.88	**30.11**
	0.8673	0.8655	0.8675	0.8760	0.8784	0.8802	0.8784	0.8621	0.8792	**0.8820**
	0.9338	0.9350	0.9379	0.9414	0.9426	0.9459	0.9426	0.9362	0.9450	**0.9468**



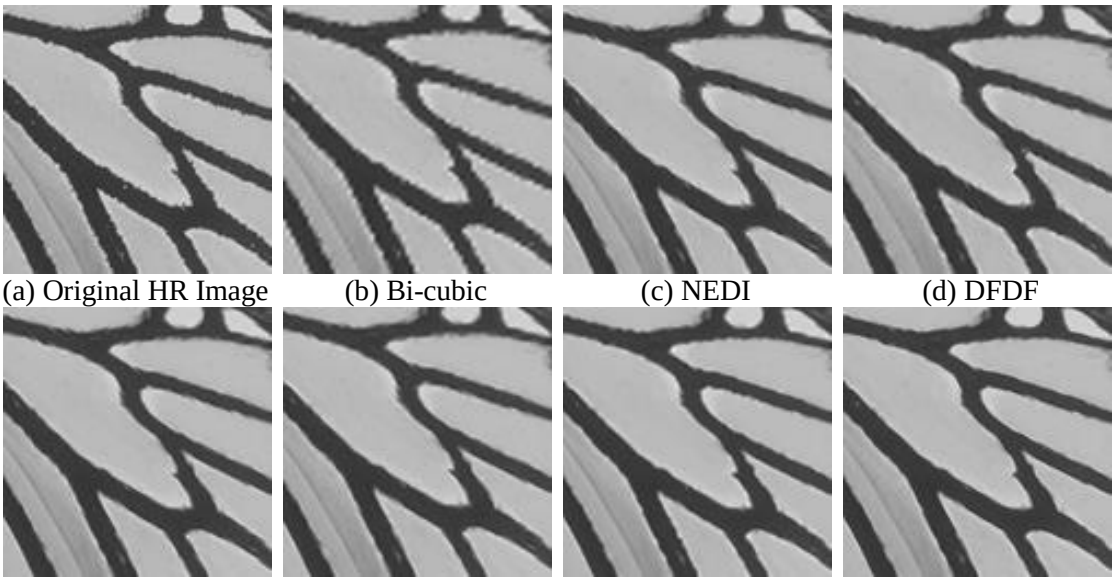
(a) Original HR Image (b) Bi-cubic (c) NEDI (d) DFDF

(e) RLLR (f) RSAI (g) NARM (h) Proposed FIRF(3,2)

Fig. 3-8: Reconstructed HR images of *Bicycle* by different interpolation methods. (a) Original HR image. (b) Bi-cubic (PSNR= 20.27dB, SSIM= 0.7559, FSIM= 0.9236). (c) NEDI (PSNR= 20.81dB, SSIM= 0.7761, FSIM= 0.9260). (d) DFDF (PSNR= 20.39dB, SSIM= 0.7606, FSIM= 0.9233). (e) RLLR (PSNR= 20.70dB, SSIM= 0.7877, FSIM= 0.9286). (f) RSAI (PSNR= 21.17dB, SSIM= 0.7904, FSIM= 0.9313). (g) NARM (PSNR= 20.46dB, SSIM= 0.7837, FSIM= 0.9282). (h) Proposed FIRF(3,2) (PSNR= **21.79**dB, SSIM= **0.8022**, FSIM= **0.9365**).

Table 3-5 summarizes the PSNR, SSIM and FSIM of the proposed FIRF(1,1), FIRF(3,2) and other methods. The average PSNR of the proposed FIRF(1,1) method is 1.09, 1.29, 0.90, 0.45, 0.30, 0.08, 0.34 and 1.25 dB higher than that of the Bi-cubic, NEDI, DFDF, RLLR, RSAI, NARM, BSAI and ICBI, respectively. Furthermore, the proposed FIRF(3,2) provides 0.23 dB improvement in PSNR over FIRF(1,1) and achieves the highest average PSNR, SSIM and FSIM compared with all the other methods.

The NARM method has 4 test images with higher PSNR compared with the proposed FIRF(3,2) method. Let us elaborate our observations. For the images containing many repetitive patterns (e.g. *Foreman* and *Fence*), the NARM method takes the advantage of exploring the nonlocal similarity across the input image which helps recover the down-scaled patterns, while the nonlocal similarities are not explored in this chapter. For the



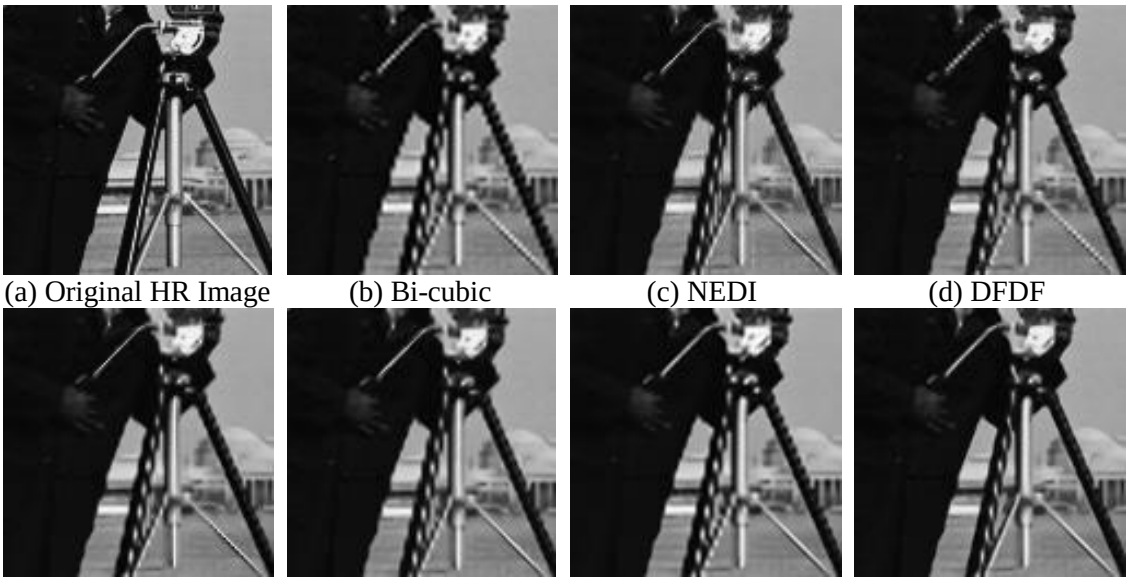
(a) Original HR Image (b) Bi-cubic (c) NEDI (d) DFDF

(e) RLLR (f) RSAI (g) NARM (h) Proposed FIRF(3,2)

Fig. 3-9: Reconstructed HR images of *Butterfly* by different interpolation methods. (a) Original HR image. (b) Bi-cubic (PSNR= 27.70dB, SSIM= 0.9241, FSIM= 0.9156). (c) NEDI (PSNR= 27.35dB, SSIM= 0.9321, FSIM= 0.9282). (d) DFDF (PSNR= 28.66dB, SSIM= 0.9397, FSIM= 0.9451). (e) RLLR (PSNR= 29.29dB, SSIM= 0.9480, FSIM= 0.9486). (f) RSAI (PSNR= 29.22dB, SSIM= 0.9472, FSIM= 0.9473). (g) NARM (PSNR= 30.35dB, SSIM= 0.9559, FSIM= 0.9571). (h) Proposed FIRF(3,2) (PSNR= **30.65**dB, SSIM= **0.9565**, FSIM= **0.9630**).

images whose smooth areas contain some noises (e.g. *Lena* and *Girl*), the slightly weaker performances of the proposed FIRF method should be owing to the lack of accuracy for the Bi-cubic interpolation to process the noisy areas. Nevertheless, our main purpose of adopting Bi-cubic interpolation to process the smooth areas is to retain proper textures and avoid over-smoothing. From the subjective viewpoint, the proposed FIRF method performs better in terms of reserving rich and natural textures.

In Figs. 3-8 to 3-10, we demonstrate the subjective performance of the proposed FIRF(3,2) method and other methods. From the subjective quality comparisons shown in these figures, we can find that the strong edges in NEDI interpolated images remain sharp, but the small edges seem unnatural, since there are some minor deformations due to the inaccurate estimation of edge directions. The overall performances of DFDF are better than that of NEDI; however, the DFDF is not good at recovering small scale edges and



(a) Original HR Image (b) Bi-cubic (c) NEDI (d) DFDF

(e) RLLR (f) RSAI (g) NARM (h) Proposed FIRF(3,2)

Fig. 3-10: Reconstructed HR images of *Cameraman* by different interpolation methods. (a) Original HR image. (b) Bi-cubic (PSNR= 25.37 dB, SSIM= 0.8627, FSIM= 0.9036). (c) NEDI (PSNR= 25.42 dB, SSIM= 0.8626, FSIM= 0.9059). (d) DFDF (PSNR= 25.67 dB, SSIM= 0.8670, FSIM= 0.9143). (e) RLLR (PSNR= 25.75 dB, SSIM= 0.8714, FSIM= 0.9156). (f) RSAI (PSNR= 26.00 dB, SSIM= 0.8730, FSIM= 0.9163). (g) NARM (PSNR= 25.92 dB, SSIM= 0.8760, FSIM= 0.9201). (h) Proposed FIRF(3,2) (PSNR= **26.57**dB, SSIM= **0.8795**, FSIM= **0.9262**).

tends to produce artifacts. The RLLR method can generate sharp edges, but sometimes the resultant edges are too sharp to be considered natural. The RSAI method and the NARM method are stable for most cases, while the NARM results seem more sharp and natural. A general problem of their algorithms is that they are not efficient enough to show some fine details in the interpolated image. Compared with the other methods, the proposed FIRF method can deal with both large scale and small scale edges effectively and is able to provide the HR images free of unnatural patterns.

It is interesting to note that Dong *et al*. [58] compared their interpolation method, NARM, with the state-of-the-art image SR method Sparse Coding Super-Resolution (ScSR) [13] and found that the NARM method achieves 1.51 dB PSNR higher than the

TABLE 3-6
AVERAGE COMPUTATIONAL TIME (MS) OF DIFFERENT INTERPOLATION METHODS

	Bicubic	**NEDI [20]**	**DFDF [24]**	**RLLR [28]**	**RSAI [27]**	**NARM [33]**	**BSAI [26]**	**ICBI [29]**	**FIRF(1,1) Proposed**	**FIRF(3,2) Proposed**
Computational Time (ms)	1.28 (C++)	3672.88 (Matlab)	2271.80 (Matlab)	32120.40 (Matlab)	3445.89 (C++)	277206.21 (Matlab)	42.55 (C++)	1510.56 (Matlab)	87.50 (C++)	384.78 (C++)
Approximated C++ Time (ms)	1	367	227	3212	3446	27721	43	151	88	385

ScSR method. Yet, our method as compared in this chapter is 0.31 dB better than NARM's approach.

3.5.4. COMPUTATIONAL EFFICIENCY

We have obtained the computational time of the un-optimized C++ implementation of the proposed FIRF method and other methods on an Intel Core i7 3.5GHz processor. In order to avoid the impression of biasing on realization, we made use of the realization codes provided by the authors on their websites for other algorithms on comparison. Some of these methods were implemented in Matlab, while the others were implemented in C++ by the respective authors. To make possible comparison, let us assume that the Matlab implementation is 10 times slower than that of the C++ implementation.

Besides providing the leading objective and subjective image interpolation results, the proposed FIRF method has significant advantage in computational efficiency. As shown in Table 3-6, the speed of the proposed FIRF(1,1) and FIRF(3,2) method is about 320 and 70 times faster than that of the NARM method. Furthermore, random forests can be parallelized. By using parallel programming, the proposed FIRF can easily give real-time performance.

Fig. 3-11 compares the computational efficiency of the image interpolation methods with respect to the approximated C++ computational time (ms) and PSNR (dB). We can observe that the proposed FIRF method achieves the highest computational efficiency.

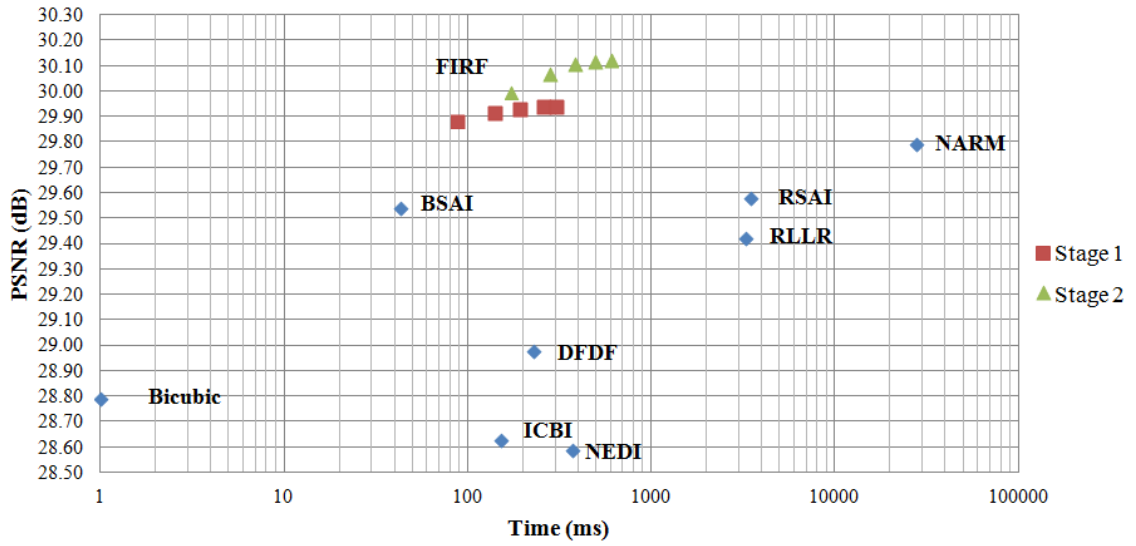
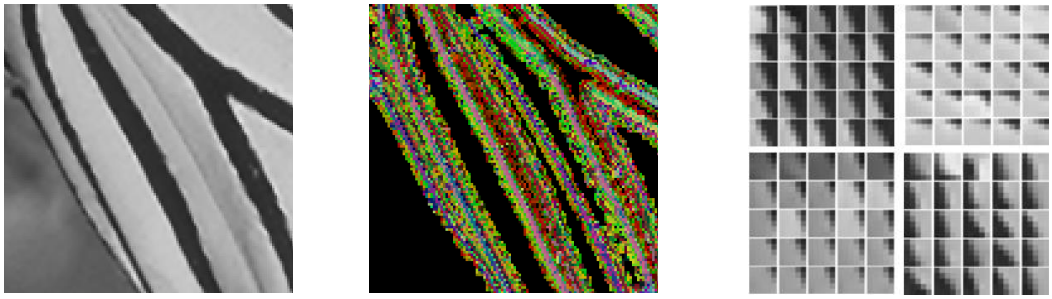


Fig. 3-11: Comparison of the average computational efficiency of the proposed FIRF(n, k) method (n=1,…, 5; k=1, 2) and other image interpolation methods.

The proposed FIRF method obtains a great PSNR gain over Bi-cubic interpolation, while only takes tens to hundreds times computational load. With comparable computational load, the average PSNR of the proposed FIRF(3,2) method is 1.52, 1.13, 1.48 and 0.53 dB higher than that of the NEDI, DFDF, ICBI and BSAI, respectively. Furthermore, from Fig. 3-11, we can find that the proposed FIRF can generate a scalable performance by varying the number of decision trees inside the random forests and using different stages for image interpolation.

The proposed FIRF method has very low computational load which mainly consists of three parts, i.e. the image patch classification process, regression models copying process and HR image patch generation process. The image patch classification process takes around 10% of the computational time. The regression models copying process and the HR image patch generation process cost about 60% of the computational time. Since the image patch classification process holds a very small portion of the computational time, the proposed method achieves very high efficiency during interpolation.



(a) Stage 1 interpolated image (b) The classification results (c) Sample classification results

Fig. 3-12: Example classification results of a Stage 2 single decision tree. (a) Stage 1 interpolated image. (b) Classification results of single decision tree in Stage 2. (c) Sample classification results from 4 leaf nodes.

3.5.5. FURTHER OBSERVATIONS

The decision tree naturally classifies an input patch into one of the leaf nodes. Fig. 3-12 (a) shows an example of the Stage 1 image interpolation result. Fig. 3-12 (b) shows the Stage 2 decision tree classification result where the pixels with the same color represent the patch centers of the same class. Basically, the pixels with similar local patch appearance are correctly marked by the same color. Fig. 3-12 (c) shows the patches which are classified into 4 leaf nodes by the Stage 2 decision tree. The patches belonging to the same leaf node share very high degree of similarity.

The nonlocal means is a well-known approach in image interpolation; however, as the similar patch searching and matching processes are very time consuming, many methods merely apply the nonlocal means within the local searching windows. With the help of the classification results of the decision trees, a patch can easily find its similar patches over the whole image, thus the true nonlocal means may be realized.

In the future, the classification property of the random forests can be further explored to enhance the image interpolation results of the proposed FIRF method. Or, the

classification results of the random forests can corporate with the nonlocal means methods to facilitate fast image interpolation.

3.6. CHAPTER SUMMARY

In this chapter we have proposed a fast image interpolation via random forests (FIRF) method. The proposed FIRF method realizes state-of-the-art image interpolation results and requires low computational load. By effectively separating the whole training data into numerous leaf nodes, the natural patch space can be represented successfully by a number of linear regression models. The simple binary tests at non-leaf nodes enable the proposed FIRF method to efficiently find suitable linear regression models for the input patch. The proposed two-stage framework generates artifacts-free image interpolation results and achieves a further 0.2 dB PSNR improvement over Stage 1 image interpolation scheme. Furthermore, the proposed FIRF is a computational scalable image interpolation method. By varying the number of decision trees and the stages applied, FIRF can adapt to devices with different computational power while giving outstanding image interpolation quality. Extensive experimental results show that our proposed FIRF method outperforms the state-of-the-art methods in terms of both objective and subjective quality and achieves the highest computational efficiency. The proposed FIRF method can generate much better results as compared to the state-of-the-art method NARM while is about 70 to 320 times faster.

CHAPTER 4 LEARNING HIERARCHICAL DECISION TREES FOR SINGLE IMAGE SUPER-RESOLUTION

4.1 INTRODUCTION

In chapter 3, we proposed fast image interpolation via random forests (FIRF) method which uses random forests to interpolate the LR image. This random forests approach is an ensemble of decision trees. Each decision tree in the random forests classifies the input LR patch into one of the leaf nodes. The retrieved regression models from the reached leaf nodes are combined to form a more robust regression model for HR patch prediction. The classification time of a decision tree has a linear relationship with its depth which is approximately the logarithm of the number of leaf nodes. This great property of the decision tree makes it favorable for fast learning-based image up-sampling.

The rest of the chapter is organized as follows: Section 4.2 introduces the proposed fast image Super-Resolution using Decision Tree (SRDT) method. Section 4.3 presents the proposed image Super-Resolution with Hierarchical Decision Trees (SRHDT) method. Section 4.4 demonstrates the data dependent model for image SR using the proposed SRDT method. Section 4.5 presents and analyzes the results of our extensive experimental work and Section 4.6 draws a conclusion.

4.2. IMAGE SUPER-RESOLUTION USING DECISION TREE

In this work, we propose to use decision tree and its variations for SR. The number of leaf nodes in a decision tree is exponential to the depth of the decision tree, and the classification time is in linear relationship to the depth of the decision tree. With this good property, the decision tree is better than the *K*-means clustering and using the Sparse

Coding to perform classification for SR. The tradeoff between image SR quality and the number of regression models can be relieved.

The general idea of image Super-Resolution using Decision Tree (SRDT) method is to efficiently classify an input LR patch into one of the leaf nodes and use the corresponding regression model to super-resolve the LR patch into its desired HR patch.

4.2.1. LEARNING THE SRDT

During training, the LR image is firstly up-sampled to the same size as the original HR image using bi-cubic interpolation. As the bi-cubic interpolation image has the same size of the HR image, we call it the $\mathbf{H}^0$ image. The root node is initialized with all the extracted training LR-HR patch pairs. For each non-leaf node with sufficient training data, we try to find an appropriate binary test which achieves the highest non-negative error reduction among a set of candidate binary tests to split the training data into its left or right child node. However, when the appropriate binary test cannot be found or the number of training data at that node is less than two times the minimum number of training data in a leaf node $2\times N_{min}$, this node will be declared as a leaf node. A non-leaf node stores the learned binary test and a leaf node store the learned regression model.

4.2.1.1. FEATURE FOR REGRESSION

The quality of super-resolved image and the running speed are in relationship with the selected LR feature for regression. The LR feature in Zeyde's method [24], ANR method [28] and A+ method [36] is the first and the second order gradient of the LR patch which dimensionality has been reduced using PCA by preserving 99.9% energy. Although multiple features can provide better estimation results, the computational complexity is

TABLE 4-1
ALGORITHM 1 SRDT TRAINING ALGORITHM

Input: HR training images **H** and LR training images **L**
1. The initially super-resolved image of **L** is the bi-cubic interpolated image $\mathbf{H}^0$
2. Extract training LR-HR patch pairs from $\mathbf{H}^0$ and **H** and initiate the root node with all the training data
3. **For** each unprocessed non-leaf node *j*
  4. **If** the training data size $N_j < 2 \times N_{min}$
    5. Declare this node as leaf node and learn the regression model $\mathbf{C}_j$
  Else
    6. Randomly generate *Q* binary tests which fulfill Eq. (4-7)
    7. **If** the highest error reduction is positive
      8. Split the training data at node *j* into its left and right child node using the learned binary test θ_j
      9. Declare this node as non-leaf node and store the learned binary test θ_j
    Else
      10. Declare this node as leaf node and learn the regression model $\mathbf{C}_j$
    End if
  End if
End for
Output: Super-Resolution Decision Tree *DT*

still much higher than using intensity feature only for estimation. The PCA transformation takes a large proportion of the overall computation time. The simple function method in [31] uses the normalized LR patch as LR patch feature. However, patch normalization is not efficient, as the patch mean intensity has to be extracted from the input LR patch and then added back to the estimated HR patch. As this chapter aims at fast image SR, we directly use the intensity values of the LR patch as the feature for HR patch estimation to simplify feature extraction process and reduce the complexity of inference. The training data are in the form of LR-HR patch pairs $\{(\mathbf{x}, \mathbf{y})\}$, where $\mathbf{x} \in R^l$ is the vectorized LR patch sampled from the bi-cubic interpolated image $\mathbf{H}^0$ and $\mathbf{y} \in R^h$ is the corresponding vectorized HR patch sampled from the original HR image. Since the bi-cubic interpolation works very well on the smooth regions, only the patches with edge pixels will be processed. The edge pixels are determined by Canny edge detector with a threshold of 20.

TABLE 4-2
ALGORITHM 2 SRDT TESTING ALGORITHM

Input: Low-resolution image **L**
1. The initially super-resolved image of the input LR image **L** is the bi-cubic interpolated image, $\mathbf{H}^0$
2. Extract LR patches from $\mathbf{H}^0$
3. **For** each LR patch **x**
   4. **For** each non-leaf node *j*
      5. Partition **x** to left or right child node according to the learned binary test θ_j, until reach a leaf node
   End for
   6. Predict the HR patch using the regression model **C** at the leaf node as Eq.(4-8)
End for
7. Reconstruct the final super-resolved image **H** by overlapping the predicted HR patches
Output: Super-resolved image **H**

4.2.1.2. LEARNING

To adapt to varying intensity scale, the binary test adopted in this chapter is specified by a set with three parameters, $\theta = \{P_1, P_2, \tau\}$. The first two parameters P_1 and P_2 represent two positions on the vectorized LR patch, and τ is a threshold value. To achieve sufficient randomness, all the parameters of the binary test are randomly generated. The difference of the intensity values between positions P_1 and P_2 is invariant to intensity changes. This kind of binary test can classify patches which may have different mean intensity but with similar appearance into the same leaf node. The split function $h(\mathbf{x}, \theta)$ checks the vectorized LR patch **x** according to the binary test θ and returns 0 when the requirement fulfills and vice versa.

$$h(\mathbf{x},\theta) = \begin{cases} 0, \text{if } \mathbf{x}(P_1) < \mathbf{x}(P_2) + \tau \\ 1, \quad \text{otherwise.} \end{cases} \tag{4-1}$$

A binary test tries to exclusively divide the training data at current node into two child nodes according to the result of split function (training data with return value 0 will be mapped to the left child node otherwise to the right child node). The goodness of a binary

test is evaluated by the amount of error reduction between the parent node and its child nodes. The error reduction R_j at node j is the difference between the fitting error at current node E_j and the weighted fitting error at its two child nodes E_L and E_R.

$$R_j = E_j - \sum_{i=\{L,R\}} \frac{N_i}{N_j} E_i, \tag{4-2}$$

where N_j is the number of training data at node j and N_i, $i=\{L, R\}$ is the number of training samples at the left or the right child node. The fitting error E_j at node j is measured by the mean squared error between the original HR patches $\mathbf{Y} \in R^{N_j} \times R^h$ and the reconstructed HR patches $\mathbf{Y}^R \in R^{N_j} \times R^h$, where N_j is the number of training samples at node j.

$$E_j = \frac{1}{N_j} \| \mathbf{Y} - \mathbf{Y}^R \|_2^2 . \tag{4-3}$$

At node j, a regression model $\mathbf{C}_j \in R^l \times R^h$ is learned using regularized linear regression to model the relationship between HR patches $\mathbf{Y}$ and LR patches $\mathbf{X} \in R^{N_j} \times R^l$.

$$\mathbf{C}_j = \arg\min_{\mathbf{C}_j} \| \mathbf{Y} - \mathbf{C}_j \mathbf{X} \|_2^2 + \lambda \| \mathbf{C}_j \|_2^2, \tag{4-4}$$

where λ is the regularization parameter to enhance the generalization ability of $\mathbf{C}j$.

Eq. (4-4) has a closed form solution.

$$\mathbf{C}_j = \mathbf{X}\mathbf{Y}^T (\mathbf{X}\mathbf{X}^T + \lambda \mathbf{I})^{-1}. \tag{4-5}$$

The reconstructed HR image patches $\mathbf{Y}^R$ are predicted using the learned regression model.

$$\mathbf{Y}^R = \mathbf{C}_j \mathbf{X}. \tag{4-6}$$

In Fast Image Interpolation via Random Forests (FIRF) method [37], a constraint on the training data size of two child nodes was proposed to improve quality of the learned decision trees. In this chapter, we adopt this setting. All the binary split should fulfill Eq. (4-7).

$$\max(N_L, N_R) \times k \le \min(N_L, N_R), \tag{4-7}$$

where the constraint parameter $k = 0.7$ follows the setting in FIRF, and N_L and N_R are the training data sizes of the left and right child node respectively.

At node j, the binary test with the highest error reduction from the Q randomly generated binary tests $\Theta = \{\theta_i \mid i = 1,..., Q\}$ is selected as the learned binary test θ_j. If no positive error reduction is achieved, that node will not be further divided and will be declared as a leaf node. By recursively partition the training LR-HR patch pairs at non-leaf nodes into leaf nodes, the SRDT can be gradually constructed. The training algorithm of the proposed SRDT method is summarized in Table 4-1 **Algorithm 1**.

4.2.2. SUPER-RESOLUTION WITH LEARNED SRDT

Table 4-2 **Algorithm 2** depicts the testing algorithm of the proposed SRDT method. Each patch with edge pixels $\mathbf{x}$ in the bi-cubic initially up-sampled image is recursively partitioned into left or right child node according to the result of the split function $h(\mathbf{x}, \theta)$ with the learned binary test until it reaches a leaf node. The reconstructed HR patch $\mathbf{y}^R$ is obtained by multiplying the LR patch $\mathbf{x}$ with the reached regression model $\mathbf{C}$ as shown in Eq. (4-8). Neighboring reconstructed HR patches are overlapped to form the final super-resolved image.

$$\mathbf{y}^R = \mathbf{C}\mathbf{x}. \tag{4-8}$$

It is good to have all kinds of LR-HR patch pairs with various appearances for training. We find that, for a leaf node ($leaf_0$) in the learned SRDT, there are three relevant leaf nodes $leaf_1$, $leaf_2$ and $leaf_3$ where the reached training LR-HR patch pairs are approximately the rotated versions (with an angle of 90°, 180° and 270°, respectively) of that at $leaf_0$. Thus after transformation, the regression models in $leaf_0$, $leaf_1$, $leaf_2$ and $leaf_3$ can have very

similar coefficients. Based on this observation, we propose two ideas to further improve the efficiency and effectiveness of our proposed SRDT method.

4.2.2.1. RESERVE A QUARTER OF THE LEAF NODES

Removing 75% of the leaf nodes in a learned SRDT makes the SRDT only require 25% of its original size. This could be significant for practical applications. However, the correspondences between leaf nodes in a decision tree are not always one-to-one. Thus, we usually cannot remove leaf nodes in a simply way. This could be a future research direction.

4.2.2.2. FUSE REGRESSION MODELS FOR TESTING

To achieve a better SR quality, combining relevant regression models in a single SRDT is a promising approach as random forests has been proven to be more robust than decision tree. For each vectorized LR patch $\mathbf{x} \in R^l$ with dimensionality of l, it can be transformed into its 90°, 180° and 270° rotated versions $\mathbf{x}_{90}$, $\mathbf{x}_{180}$ and $\mathbf{x}_{270}$ respectively.

$$\mathbf{x}_{90}(i) = \mathbf{x}(l - \sqrt{l} \times (i \% \sqrt{l} + 1) + i / \sqrt{l}). \tag{4-9}$$

$$\mathbf{x}_{180}(i) = \mathbf{x}(l - 1 - i). \tag{4-10}$$

$$\mathbf{x}_{270}(i) = \mathbf{x}(\sqrt{l} \times (i \% \sqrt{l} + 1) - i / \sqrt{l} - 1). \tag{4-11}$$

$\mathbf{x}$, $\mathbf{x}_{90}$, $\mathbf{x}_{180}$ and $\mathbf{x}_{270}$ reach 4 different leaf nodes in the learned SRDT. The retrieved regression models $\mathbf{C}_{90}$, $\mathbf{C}_{180}$ and $\mathbf{C}_{270}$ by $\mathbf{x}_{90}$, $\mathbf{x}_{180}$ and $\mathbf{x}_{270}$ will be manipulated into the form that is suitable to super-resolve patch $\mathbf{x}$.

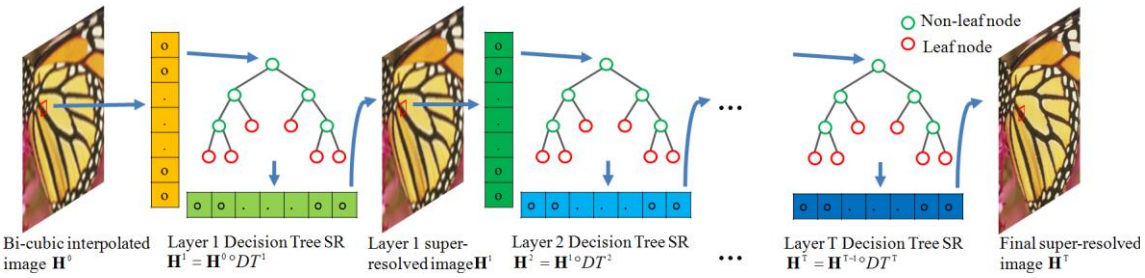


Fig. 4-1: Flowchart of the proposed image super-resolution using hierarchical decision trees method.

$$\mathbf{C}_{90}^{M}(l-\sqrt{l}\times(i\%\sqrt{l}+1)+i/\sqrt{l}, l-\sqrt{l}\times(j\%\sqrt{l}+1)+j/\sqrt{l}) = \mathbf{C}_{90}(i,j). \tag{4-12}$$

$$\mathbf{C}_{180}^{M}(l-1-i, l-1-j) = \mathbf{C}_{180}(i,j). \tag{4-13}$$

$$\mathbf{C}_{270}^{M}(\sqrt{l}\times(i\%\sqrt{l}+1)-i/\sqrt{l}-1, \sqrt{l}\times(j\%\sqrt{l}+1)-j/\sqrt{l}-1) = \mathbf{C}_{270}(i,j). \tag{4-14}$$

For HR patch prediction, the regression model used in Step 6 of **Algorithm 2** is the average of $\mathbf{C}$, $\mathbf{C}_{90}^{M}$, $\mathbf{C}_{180}^{M}$ and $\mathbf{C}_{270}^{M}$.

$$\mathbf{y}^{R} = \frac{1}{4}(\mathbf{C}+\mathbf{C}_{90}^{M}+\mathbf{C}_{180}^{M}+\mathbf{C}_{270}^{M})\times\mathbf{x}. \tag{4-15}$$

4.3. IMAGE SUPER-RESOLUTION USING HIERARCHICAL DECISION TREES

Although a large number of leaf nodes reduces ambiguity and improves the HR patch prediction accuracy, the training efficiency of SRDT method drops as the training data size increases. Using more training data to obtain a bigger super-resolution decision tree may not be an efficient and effective way to get better image SR quality. Besides, the size of the learned SRDT is exponentially increasing as the depth of the decision tree ascends. This could hinder the feasibility of the proposed algorithm.

In SRRF method [96], random forests approach which is an ensemble of decision trees is adopted for image SR. By combining the regression models from multiple de-correlated decision trees, a more robust regression model can be obtained for HR patch prediction.

However, the improvement of the SR quality is minor compared with the increment of computational time. In SRRF method [96], with one more decision tree in the random forests, the average computational time increases about 70 ms, while the computational time of using a single decision tree is around 100 ms and the saturated PSNR of the SR random forests with 4 decision trees is only less than 0.1 dB higher than a single decision tree. From both efficiency and effectiveness aspects, we cannot get further improvement (in the direction for the exploitation of fast and high quality image SR algorithm) from random forests structure.

We propose, in this chapter, to perform Super-Resolution using a Hierarchical Decision Trees (SRHDT) framework to further boost the image SR quality of the proposed SRDT method with short computational time. Fig. 4-1 presents the flow diagram of the proposed SRHDT method. The general idea is that each layer of the SRHDT framework pushes the estimated HR patch closer to the ground-truth HR patch. The SRHDT method progressively refines the initial bi-cubic interpolated image using hierarchical decision trees $HDT = \{DT^1, ..., DT^T\}$, and each refinement is performed by adding one more layer of decision tree DT^t which is trained with the LR-HR patch pairs from the previous layer super-resolved images and the original HR images for a set of training images. Layer *i* decision tree predicts the HR image $\mathbf{H}^i$ and is denoted as:

$$\mathbf{H}^i = \mathbf{H}^{i-1} \circ DT^i. \tag{4-16}$$

The operation '$\circ$' consists of LR patch extraction, HR patch prediction and HR image reconstruction corresponding to **Algorithm 2** SRDT method testing algorithm. The learned SRDT in each layer can also follow the regression model fusion strategy as

TABLE 4-3
ALGORITHM 3 SRHDT TRAINING ALGORITHM

Input: There are M sets of HR training images $\mathbf{H}_m$ and LR training images $\mathbf{L}_m$, for $m=\{1,\ldots, M\}$

1. The initially super-resolved image of input $\mathbf{L}_m$ is the bi-cubic interpolated image $\mathbf{H}_m^0$
2. Train Layer 1 Super-Resolution Decision Tree DT^1 using $\mathbf{L}_1$ and $\mathbf{H}_1$
3. **For** m=2 to M
   4. **For** i=1 to m-1
      5. $\mathbf{H}_m^i = \mathbf{H}_m^{i-1} \circ DT^i$
   End for
   6. Train the Layer m Super-Resolution Decision Tree DT^m using $\mathbf{H}_m^{m-1}$ and $\mathbf{H}_m$

End for

Output: Hierarchical Decision Trees $HDT = \{DT^1, \ldots, DT^T\}$

TABLE 4-4
ALGORITHM 4 SRHDT TESTING ALGORITHM

Input: Low-resolution image **L**

1. The initially super-resolved image of input LR image **L** is the bi-cubic interpolated image $\mathbf{H}^0$
2. **For** t=1 to T
   3. Super-resolve image $\mathbf{H}^t = \mathbf{H}^{t-1} \circ DT^t$

End for

Output: Super-resolved image $\mathbf{H}^T$

described in Section 4.2.2.2 by which the desired HR patch of each LR patch is predicted according to the fused regression model using 4 relevant regression models in a decision tree for higher accuracy. We denote the SRHDT method with fused regression model as SRHDT_f.

The training procedure of SRDT in each layer in the SRHDT follows the description of **Algorithm 1** using the LR patch from the super-resolved images applied by the previous layer decision trees and the HR patches from the original HR training images. The training algorithm of the SRHDT method is shown in Table 4-3 **Algorithm 3**.

With the learned hierarchical decision trees $HDT = \{DT^1, \ldots, DT^T\}$, the initially bi-cubic interpolation input LR image $\mathbf{H}^0$ will be enhanced layer by layer. Table 4-4

Algorithm 4 shows the testing algorithm of the proposed SRHDT method.

The proposed SRHDT is similar to the spirit in deep learning based image SR method, SRCNN which has linear transformation and nonlinear mapping. In SRHDT, the HR image patch prediction performed on a leaf node is a linear transformation and the patch overlapping is a nonlinear mapping which clips the dynamic range of pixel intensity between 0 and 255. Multiple layers in SRHDT could correspond to multiple linear transformation and nonlinear mapping structure in deep learning model.

4.4. DATA DEPENDENT MODEL FOR VIDEO SUPER-RESOLUTION

For single image SR, it is difficult to learn a complete model which can provide superb SR quality for variant image contents. Without the prior knowledge of the image content, the ambiguity between LR patches in training images impedes us to get a better SR results. Video SR is in a different scenario as single image SR. There are multiple similar images of the same scene in a video. If a few HR images (for example, I frames) are provided for each scene, a data dependent model can learn from the provided LR-HR image pairs. With the learned data dependent model, the desired HR images of the input LR images (for example, B frames) within the same scene can be predicted more effectively.

Applying image SR technique for video compression and video coding is a promising research direction. However, these applications require the minimization of the difference between the original HR images and the super-resolved images. The existing general image SR models are not good enough to accomplish this task. Designing a learning-based SR method with data dependent model could be one of the possible solutions.

All the learning-based SR methods including the proposed SRDT method and SRHDT method can be converted into a dependent model using the provided LR-HR image pairs.

Different from the general model which can be learned offline without considering the training time, the training speed for data dependent model should be as fast as possible. In SRDT and SRHDT methods, much training time is used to construct the regression models for error reduction evaluation. In data dependent model for video SR, we propose to randomly select binary splits without error reduction evaluation which could sacrifice slightly the SR quality. However, the image SR quality using data dependent SRDT and SRHDT is much better than using the learned general SRDT and SRHDT. The only difference between the data dependent SRDT training algorithm and **Algorithm 1** is that in the data dependent algorithm if the training data (in a node) which are enough for further split it will randomly find a binary split fulfilling Eq. (4-7) to partition the training data without considering the error reduction.

The data dependent model SRDT training algorithm is around 80 times faster than the original general model SRDT training algorithm. The average PSNR only has around a 0.1 dB decrease.

4.5. EXPERIMENTAL RESULTS

In this section, we report the testing results and analyze the proposed the SRDT method, the SRHDT method as well as the SRHDT_f method with upscaling factors of 2 and 3. We adopt the Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity (SSIM) index [94] to evaluate the image SR quality for all methods. For color image, we converted the image from RGB color space to YCbCr color space (Y axis represents luminance, Cb axis and Cr axis are chrominance channels) and performed SR only on the luminance channel. The chrominance channels were up-sampled to the desired size by bi-cubic interpolation method. For upscaling factor of p, the LR image was obtained by a bi-cubic

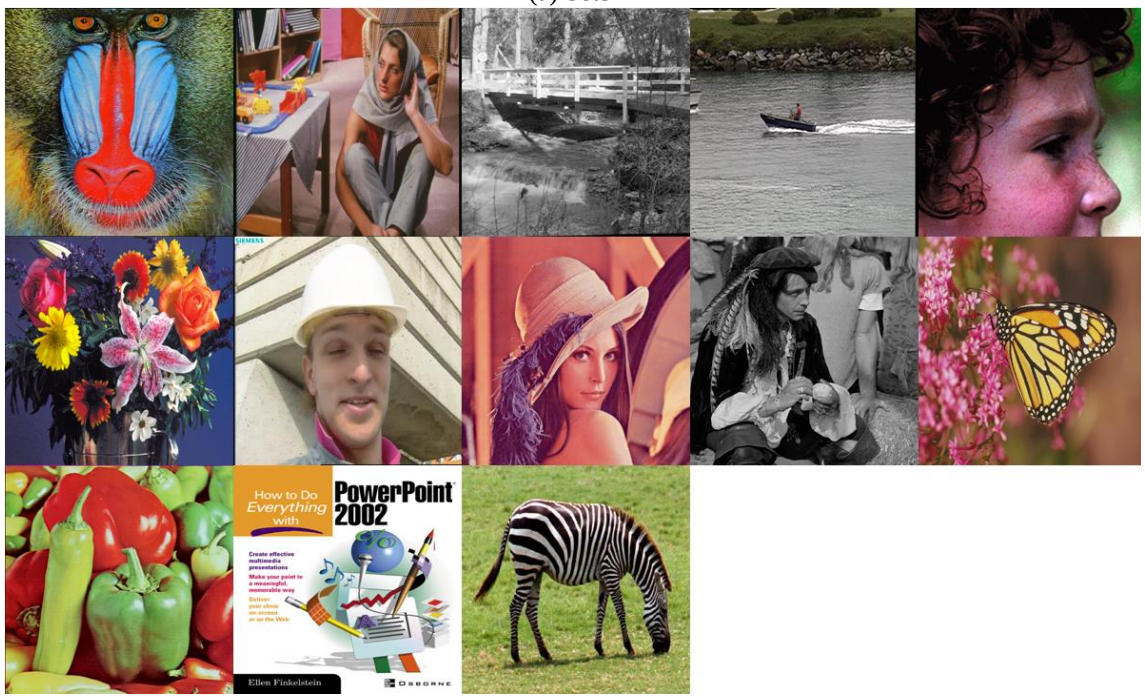
Fig. 4-2: Parts of the training images.

filter followed by a downsampling by a factor of p (realized by MATLAB function *imresize*); the HR patch size was selected as $(3p)\times(3p)$ so that the corresponding LR patch size is always 3×3. All the experimental results were obtained on an Intel Core i7 3.5GHz processor. The proposed methods were implemented in C++, taking the Bi-cubic interpolation image as input.

4.5.1. SINGLE IMAGE SUPER-RESOLUTION WITH GENERAL MODEL

For the general model, we used 118 training images for training (Note that the general model refers to that the regression models which can be used to construct a super-resolution of any LR image, i.e. different sets of images are used for training and super-resolution evaluation). All the training images were downloaded from Featured pictures in Wikipedia within the title of "Plants", parts of the training images are shown in Fig. 4-2. The training images can provide around 10,000,000 LR-HR patch pairs for each layer super-resolution decision tree training. *Set5* [13] (shown in Fig. 4-3 (a), with 5 testing *images*: *baby*, *bird*, *butterfly*, *head* and *woman*) and *Set14* [24] (shown in Fig. 4-3 (b), with 14 testing images: *baboon*, *barbara*, *bridge*, *coastguard*, *comic*, *face*, *flowers*, *foreman*, *lenna*, *man*, *monarch*, *pepper*, *ppt3* and *zebra*) were used to evaluate for the image SR quality.

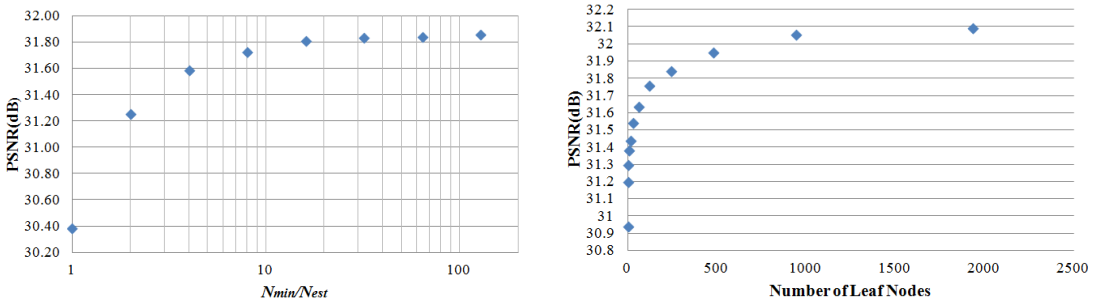
(a) Set5



(b) Set14

Fig. 4-3: Testing images: (a) Set5; (b) Set14.

4.5.1.1. EXPERIMENTAL SETTINGS



(a) (b)

Fig. 4-4: Upscaling factor = 2: (a) the relationship between N_{min} and the average PSNR of the super-resolved images in Set14 when magnification factor is 2, with the number of leaf nodes = 256; (b) the relationship between number of leaf nodes and the average PSNR of the super-resolved images in *Set14* with *Nmin* = 1800.

Performance of the SRDT is controlled by the quality of the regression models and the number of leaf nodes. The quality of the regression models can be improved when using more training LR-HR patch pairs to construct each of the regression models. The reason behind is obvious. The relationship between LR and HR patches can be more precisely represented using a denser sampled training data. Fig. 4-4 (a) shows the relationship between the minimum number of training data in a leaf node N_{min} and the average PSNR of the super-resolved images in *Set14* when the leaf node number is around 256 and the upscaling factor is 2. As N_{min} increases from 1 to 128 times of the minimum number of LR-HR patch pairs required to estimate the regression model N_{est} (where the patch size used is 6×6, the minimum number is 36), the average PSNR of the test images has around 1.4 dB improvement. In the following experiments, we adopted N_{min} as 50 times of N_{est}. The number of leaf nodes is a key factor that affects the image super-resolution accuracy. The basic idea of recent fast image SR methods [28, 31, 33, 34, 36, 38] is to divide the image patch space into numerous subspaces and use simple linear model to relate LR and HR patches. The number of leaf nodes thus affects the subspace size. If the subspace is too big, the assumption that a simple linear model can represent

TABLE 4-5

PSNR (DB), SSIM AND AVERAGE RUNNING TIME (S) RESULTS ON SET14 BY BI-CUBIC INTERPOLATION AND THE PROPOSED SRHDT METHOD WITH DIFFERENT NUMBER OF LAYERS FOR UPSCALING FACTORS X2

Images	**Methods**													
	Bi-cubic		**SRHDT 1 Layer**			**SRHDT 2 Layers**			**SRHDT 3 Layers**			**SRHDT 4 Layers**		
	PSNR	SSIM	PSNR	SSIM	Time	PSNR	SSIM	Time	PSNR	SSIM	Time	PSNR	SSIM	Time
Baboon	24.87	0.6980	25.63	0.7597	0.13	25.73	**0.7710**	0.24	25.77	0.7698	0.33	**25.78**	0.7701	0.44
Barbara	27.94	0.8398	**28.51**	0.8685	0.16	28.35	**0.8705**	0.30	28.29	0.8694	0.43	28.21	0.8689	0.57
Bridge	26.60	0.7936	27.70	0.8446	0.15	27.83	**0.8522**	0.27	27.88	0.8516	0.40	**27.89**	0.8517	0.49
Coastguard	29.21	0.7879	30.50	0.8393	0.05	30.64	**0.8480**	0.09	30.64	0.8460	0.13	**30.66**	0.8460	0.17
Comic	25.93	0.8470	28.01	0.9053	0.05	28.44	0.9157	0.09	28.60	0.9175	0.13	**28.70**	**0.9191**	0.17
Face	34.78	0.8622	35.57	0.8812	0.03	35.62	**0.8837**	0.06	**35.63**	0.8831	0.08	**35.63**	0.8829	0.11
Flowers	30.28	0.8979	32.63	0.9305	0.08	33.08	0.9355	0.16	33.26	0.9365	0.21	**33.37**	**0.9372**	0.28
Foreman	34.67	0.9535	37.60	0.9684	0.03	38.04	0.9702	0.07	**38.19**	**0.9706**	0.09	38.18	**0.9706**	0.12
Lenna	34.64	0.9109	36.40	0.9254	0.09	36.56	0.9270	0.17	36.62	**0.9271**	0.23	**36.64**	**0.9271**	0.31
Man	29.21	0.8454	30.65	0.8804	0.13	30.92	0.8865	0.24	31.00	0.8868	0.33	**31.04**	**0.8872**	0.43
Monarch	32.86	0.9598	36.80	0.9746	0.12	37.53	0.9763	0.23	37.77	0.9767	0.32	**37.90**	**0.9770**	0.42
Pepper	35.01	0.9071	36.83	0.9174	0.09	37.02	0.9185	0.16	37.09	0.9187	0.23	**37.12**	**0.9188**	0.31
Ppt3	26.79	0.9439	30.08	0.9764	0.09	30.79	0.9807	0.18	31.08	0.9822	0.26	**31.27**	**0.9830**	0.34
Zebra	30.57	0.9090	33.63	0.9412	0.12	33.68	**0.9429**	0.24	33.78	0.9427	0.32	**33.80**	0.9426	0.42
Average	30.24	0.8683	32.18	0.9009	0.09	32.45	0.9056	0.18	32.54	0.9056	0.26	**32.58**	**0.9059**	0.33

the LR-HR relationship in each subspace would no longer hold. Fig. 4-4 (b) illustrates the PSNR of the test images in *Set14* with respect to different number of leaf nodes. We vary the number of leaf nodes from 1 to around 2048 by using the number of training data varying from 2,500 to 5,120,000. It can be seen from Fig. 4-4 (b) that there is an around 0.1 dB improvement in PSNR when the number of leaf nodes (the size of training data) doubles. From SRDT perspective, although learning large enough decision tree is beneficial, an exponential increase in training time and training data would become unaffordable.

To accommodate the huge training time and training data for a big SRDT, we propose the SRHDT method. We rotated the HR training images with certain angles to generate enough training data to form multiple layers in the SRHDT method. Table 4-5 presents the PSNR, SSIM and running time of each layer in SRHDT (totally 4 layers) on *Set14* [29] for an upscaling factor of 2. The SRDT in each layer had around 4000 leaf nodes. The running time of the SRHDT method increases linearly as the increment of layer

TABLE 4-6
PSNR (DB), SSIM AND AVERAGE RUNNING TIME (S) BY DIFFERENT IMAGE SUPER-RESOLUTION METHODS FOR UPSCALING FACTORS X2 AND X3 ON SET5 AND SET14 (THE BEST RESULT IN EACH ROW IS MARKED BY RED AND THE SECOND BEST RESULT IS MARKED BY BLUE)

Dataset	*Scale*	*Eval. Mat*	**Bi-cubic**	**Zeyde's [29]**	**A+ [35]**	**SRCNN [37]**	**Peleg's [31]**	**Proposed SRDT**	**Proposed SRHDT**	**Proposed SRHDT_f**
Set5	×2	PSNR	33.66	35.79	36.56	36.35	36.11	36.32	**36.77**	**36.92**
		SSIM	0.9299	0.9494	**0.9545**	0.9522	0.9504	0.9514	0.9539	**0.9546**
		Time	0.000	2.570	0.810	5.120	9.770	**0.037**	**0.150**	0.410
	×3	PSNR	30.41	31.94	32.66	32.42	32.36	32.36	**32.94**	**33.08**
		SSIM	0.8686	0.8973	0.9095	0.9024	0.9040	0.9030	**0.9114**	**0.9132**
		Time	0.000	1.880	0.670	7.700	18.590	**0.063**	**0.256**	0.555
Set14	×2	PSNR	30.24	31.84	32.32	32.23	31.99	32.18	**32.58**	**32.67**
		SSIM	0.8683	0.8985	0.9054	0.9036	0.9005	0.9009	**0.9059**	**0.9069**
		Time	0.000	5.490	1.730	11.300	20.270	**0.084**	**0.339**	0.903
	×3	PSNR	27.54	28.67	29.16	29.03	28.95	28.97	**29.38**	**29.46**
		SSIM	0.7726	0.8067	0.8182	0.8128	0.8129	0.8127	**0.8203**	**0.8220**
		Time	0.000	2.960	1.050	9.830	44.060	**0.137**	**0.553**	1.230

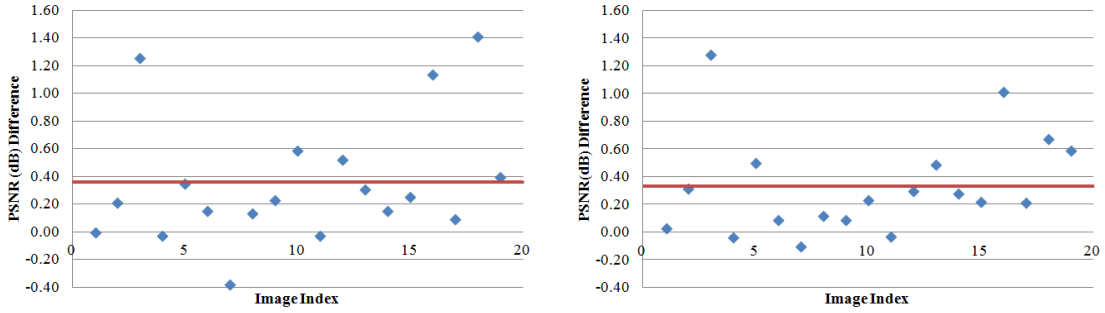
TABLE 4-7
PARAMETERS SETTING UNDER UPSCALING FACTOR OF 2 AND 3

Scenario	Patch Size	λ	*k*	*Nmin*	*Q*	T	Overlapping
×2	6×6	0.01	0.7	1800	36	4	4 pixels
×3	9×9	0.01	0.7	4020	81	4	6 pixels

number in the SRHDT. The SRHDT with 4 layers achieves the highest PSNR and SSIM on most of the testing images in *Set14*. We can find that there has a 0.26 dB, 0.10 dB and 0.04 dB improvement in PSNR for layer 2, 3 and 4 respectively. The overall PSNR improvement compared with the layer 1 result is 0.40 dB. With the same PSNR improvement, the SRDT may need around 16 times more training data (under the assumption that when training data double, there is around 0.1 dB improvement).

4.5.1.2. COMPARISON WITH STATE-OF-THE-ART ALGORITHMS

As stated in Section 4.2.2.2, our proposed hierarchical framework with fused regression model (SRHDT_f) method can achieve even better SR results by fusing regression models from relevant leaf nodes within the same decision tree. Table 4-6 reports the average PSNR, SSIM and average running time of the proposed SRDT method,

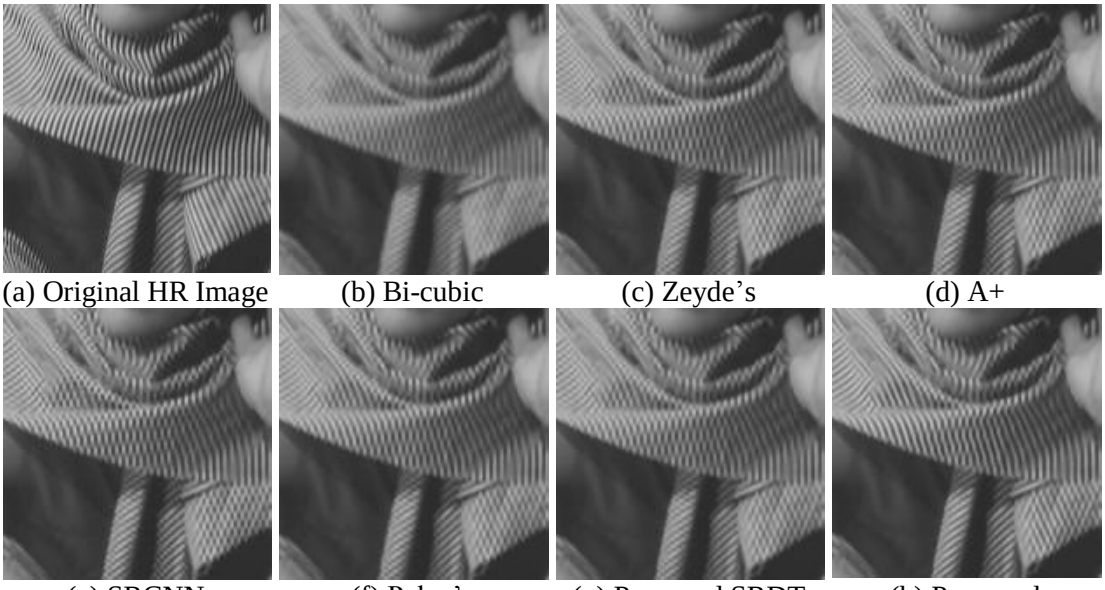


(a) (b)

Fig. 4-5: PSNR(dB) differences between the proposed SRHDT_f method and the A+ method of all the testing images in Set5 and Set14 (a) for upscaling factor of 2, with an average difference in PSNR 0.36 dB; (b) for upscaling factor of 3, with an average difference in PSNR 0.33 dB.

SRHDT method, SRHDT_f method and the state-of-the-art image SR algorithms including the Zeyde's method [24], A+ method [36], SRCNN method [34] and Peleg's method [35] for upscaling factors of 2 and 3 on *Set5* and *Set14*. There were 4 decision trees in the SRHDT method and SRHDT_f method in which the number of leaf nodes was around 4000 and 2000 for upscaling factors of 2 and 3, respectively. Among the comparison methods, Zeyde's method [24] is the state-of-the-art fast image SR algorithm which is based on orthogonal matching pursuit for sparse coding and uses PCA for input LR feature dimensionality reduction. A+ method [36] achieved the best performance in the literature on both SR quality and speed by using sparse coding for patch classification and pre-learned regression models. SRCNN method [34] explores to apply Conventional Neural Network for SR and realized with high quality and fast speed. Peleg's method [35] applies an efficient feedforward Neural Network implementation for HR patch sparse representation prediction which leads to fast realization. We have used the realization codes from the authors' websites for comparison to avoid experiment biasing.

Table 4-7 shows the setting of parameters for our proposed methods: the patch sizes, regularization parameters, constraint parameters, minimum number of training data in a



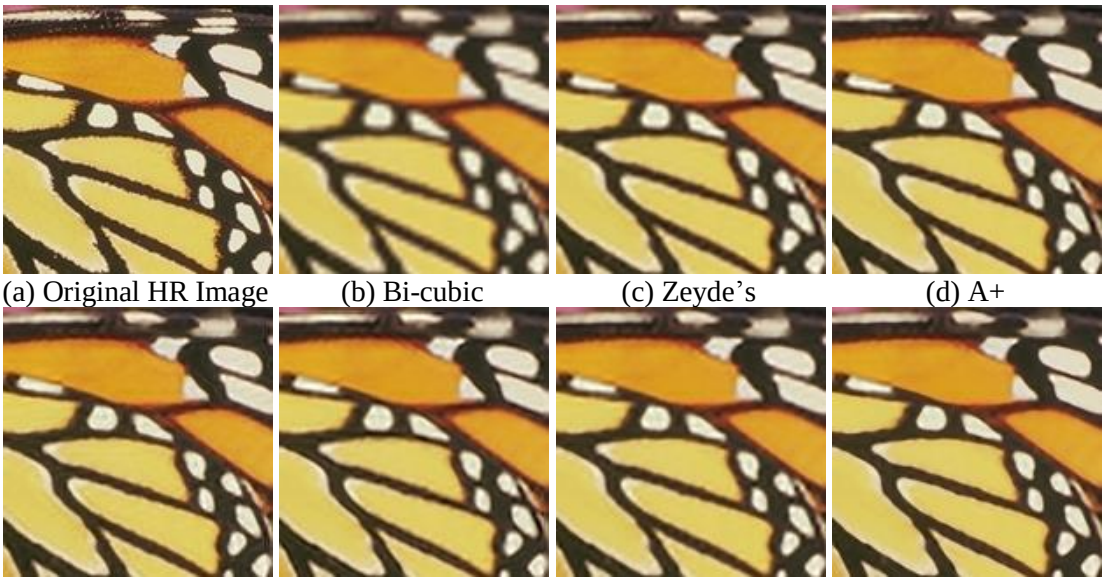
(a) Original HR Image (b) Bi-cubic (c) Zeyde's (d) A+

(e) SRCNN (f) Peleg's (g) Proposed SRDT (h) Proposed SRHDT_f

Fig. 4-6: Reconstructed HR images of *barbara* from Set14 dataset by different super-resolution methods for upscaling factor of 2. (a) Original HR image. (b) Bi-cubic (PSNR= 27.94dB, SSIM= 0.8398). (c) Zeyde's (PSNR= **28.63**dB, SSIM= 0.8717). (d) A+ (PSNR= **28.63**dB, SSIM= 0.8721). (e) SRCNN (PSNR= 28.53dB, SSIM= **0.8743**). (f) Peleg's (PSNR= 28.48dB, SSIM= 0.8688). (g) Proposed SRDT (PSNR= 28.51dB, SSIM= 0.8685). (h) Proposed SRHDT_f (PSNR= 28.25dB, SSIM= 0.8701).

leaf node, number of randomly generated binary test in a node, number of layers in hierarchical decision trees and overlapping pixels.

To summarize the quality performance in Table 4-6, let us average the results on *Set5* and *Set14*. For upscaling factor of 2, the average PSNR of the proposed SRHDT_f method are 0.10, 0.36, 0.47, 0.52, 0.72 and 0.91 dB higher than that of the proposed SRHDT method, A+ method, SRCNN method, the proposed SRDT method, Peleg's method and Zeyde's method. For upscaling factor of 3, the average PSNR of the proposed SRHDT_f method are 0.10, 0.33, 0.49, 0.55, 0.56 and 0.88 dB higher than that of the proposed SRHDT method, A+ method, SRCNN method, the proposed SRDT method, Peleg's method and Zeyde's method. The proposed regression model fusion method SRHDT_f leads the proposed SRHDT method with around 0.1 dB gain without learning extra



(a) Original HR Image (b) Bi-cubic (c) Zeyde's (d) A+

(e) SRCNN (f) Peleg's (g) Proposed SRDT (h) Proposed SRHDT_f

Fig. 4-7: Reconstructed HR images of *Butterfly* from Set5 dataset by different super-resolution methods for upscaling factor of 3. (a) Original HR image. (b) Bi-cubic (PSNR= 24.09dB, SSIM= 0.8241). (c) Zeyde's (PSNR= 26.06dB, SSIM= 0.8803). (d) A+ (PSNR= 27.46dB, SSIM= 0.9124). (e) SRCNN (PSNR= 27.76dB, SSIM= 0.9031). (f) Peleg's (PSNR= 26.92dB, SSIM= 0.9022). (g) Proposed SRDT (PSNR= 27.07dB, SSIM= 0.8995). (h) Proposed SRHDT_f (PSNR= **28.74**dB, SSIM= **0.9280**).

information. Compared to the state-of-the-art image SR methods, SRHDT_f has an around 0.3 and 0.5 dB improvement over the A+ method and SRCNN method for both upscaling factors of 2 and 3, respectively. It is surprising to find that the proposed SRDT method has comparable results with the SRCNN method and Peleg's method while has around 0.3 dB advantage over the Zeyde's method. To further demonstrate the performance of the proposed SRHDT_f method, Fig. 4-5 shows the PSNR (dB) differences between the results of the SRHDT_f method and the A+ method on all testing images for upscaling factors of 2 and 3, respectively. The proposed SRHDT_f can provide more than 1 dB gain in PSNR compared to the A+ method on testing images *butterfly*, *monarch* and *ppt3*. Although the proposed SRHDT_f method reports lower PSNR on testing image *barbara*, which could be due to the strongly aliased patterns on *barbara* as shown in Fig. 4-6.

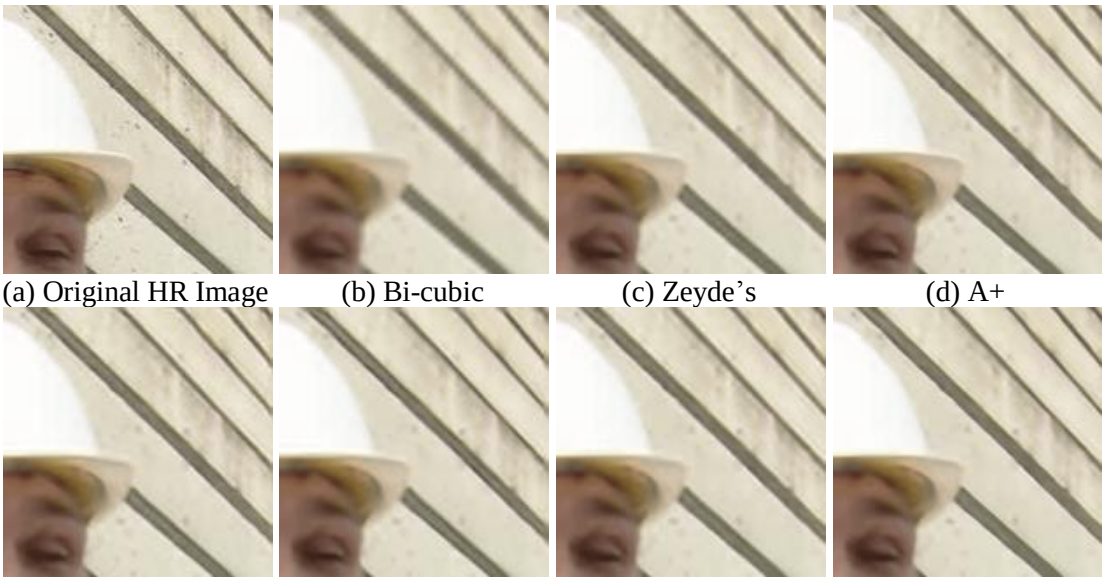
(a) Original HR Image (b) Bi-cubic (c) Zeyde's (d) A+

(e) SRCNN (f) Peleg's (g) Proposed SRDT (h) Proposed SRHDT_f

Fig. 4-8: Reconstructed HR images of *comic* from Set14 dataset by different super-resolution methods for upscaling factor of 2. (a) Original HR image. (b) Bi-cubic (PSNR= 25.93dB, SSIM= 0.8470). (c) Zeyde's (PSNR= 27.55dB, SSIM= 0.8968). (d) A+ (PSNR= 28.19dB, SSIM= 0.9118). (e) SRCNN (PSNR= 28.17dB, SSIM= 0.9099). (f) Peleg's (PSNR= 27.94dB, SSIM= 0.9045). (g) Proposed SRDT (PSNR= 28.01dB, SSIM= 0.9053). (h) Proposed SRHDT_f (PSNR= **28.78**dB, SSIM= **0.9205**).

However, our proposed SRHDT_f method generates better visual quality with sharper edges as shown in the figure.

The runtime efficiency of the proposed SRDT method, SRHDT method and SRHDT_f method is a great advantage over other methods. The proposed SRDT method is the fastest image SR algorithm in Table 4-6 except the bi-cubic interpolation method. With comparable SR quality, the proposed SRDT method requires only around 1% and 0.4% processing time of the SRCNN method and Peleg's method. The running time of the proposed SRHDT method is about 4 times longer than that of the proposed SRDT method, since there are 4 layers in the SRHDT framework. The SRHDT method achieves the second best PSNR in every experimental settings. Moreover, its running speed is 2 to 5 times faster than A+ method which obtains the third best PSNR results. Fusion regression



(a) Original HR Image (b) Bi-cubic (c) Zeyde's (d) A+

(e) SRCNN (f) Peleg's (g) Proposed SRDT (h) Proposed SRHDT_f

Fig. 4-9: Reconstructed HR images of *foreman* from Set14 dataset by different super-resolution methods for upscaling factor of 3. (a) Original HR image. (b) Bi-cubic (PSNR= 31.96dB, SS IM= 0.9099). (c) Zeyde's (PSNR= 34.16dB, SSIM= 0.9323). (d) A+ (PSNR= 35.61dB, SSIM= 0.9428). (e) SRCNN (PSNR= 34.82dB, SSIM= 0.9339). (f) Peleg's (PSNR= 35.09dB, SSIM= 0.9388). (g) Proposed SRDT (PSNR= 34.78dB, SSIM= 0.9353). (h) Proposed SRHDT_f (PSNR= **36.09**dB, SSIM= **0.9441**).

models from relevant leaf nodes in a decision tree would make the testing time of the proposed SRHDT_f method twice as the proposed SRHDT method.

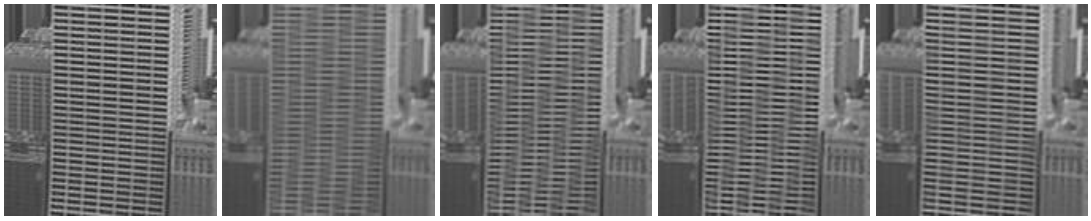
In Figs. 4-6 ~ 4-9, the subjective performances of the proposed SRDT method, SRHDT_f method and other methods are demonstrated. From visual comparisons, we can see that the reconstructed HR images of our proposed SRHDT_f method are with sharper edges and less artifacts compared to those of other state-of-the-art image SR algorithms.

4.5.2. VIDEO SUPER-RESOLUTION WITH DATA DEPENDENT MODEL

In this section, we evaluate the proposed SRDT for video SR with data dependent model using 5 video sequences *City*, *Crew*, *Harbour*, *Ice* and *Soccer* for an upscaling factor of 2 (the first image of each sequence is shown in Fig. 4-10). All these sequences are with spatial resolution of 704×576 and a frame rate of 30 fps. For each video sequence,

(a) City (b) Crew (c) Harbour (d) Ice (e) Soccer

Fig. 4-10: The first image of the testing sequence: (a) *City*, (b) *Crew*, (c) *Harbour*, (d) *Ice* and (e) *Soccer*.



(a) Original HR Image (b) Bi-cubic (c) General Model SRDT (d) General Model SRHDT_f (e) Data Dependent SRDT

Fig. 4-11: Reconstructed HR images of the 16th frame in the testing sequence *City* by different super-resolution methods for upscaling factor of 2. (a) Original HR image. (b) Bi-cubic (PSNR= 28.18dB, SSIM= 0.8336). (c) General Model SRDT (PSNR= 30.11dB, SSIM= 0.8885). (d) General Model SRHDT_f (PSNR= 31.02dB, SSIM= 0.9011). (e) Data dependent SRDT (PSNR= **31.89**dB, SSIM= **0.9012**).

the 1st frame and the 31th frame were selected as training data, and the 2nd frame to the 30th frame are testing frames. Since the training data is not sufficient, it is the first priority to obtain more leaf nodes in the learned super-resolution decision tree. The minimum number of training data in a leaf node *Nmin* was set to 2×*Nest* and the regularization parameter λ was set to 1. The SRDT method has to be evaluated not only by the quality assessment metrics (PSNR and SSIM) but also the training time. Because too long training time is unacceptable for video application which requires a huge number of frames.

Table 4-8 illustrates the average PSNR and SSIM of the Bi-cubic interpolation, the proposed general model SRDT method, the proposed general model SRHDT_f method and the proposed data dependent SRDT method (with the training time in minutes) on 5 testing video sequences. We can find that the proposed data dependent SRDT method achieves around 1.0 dB and 1.9 dB gain in PSNR compared to the proposed general model

TABLE 4-8
PSNR(DB) AND SSIM RESULTS BY BI-CUBIC INTERPOLATION, THE PROPOSED GENERAL MODEL SRDT METHOD, THE PROPOSED SRHDT_F METHOD, AND THE PROPOSED DATA DEPENDENT MODEL SRDT METHOD (WITH TRAINING TIME (MIN)) ON 5 TESTING SEQUENCES FOR UPSCALING FACTOR OF 2

Sequences	**Methods**								
	Bi-cubic		General SRDT Method		General SRHDT_f Method		Data Dependent SRDT Method		
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	Training Time
City	29.11	0.8558	31.18	0.9060	32.13	0.9170	**33.08**	**0.9179**	5.52
Crew	36.51	0.9392	38.77	0.9524	38.85	0.9540	**39.59**	**0.9547**	2.36
Harbour	31.83	0.9314	35.10	0.9631	34.76	0.9641	**35.87**	**0.9661**	6.11
Ice	35.58	0.9578	38.82	0.9682	39.41	**0.9697**	**39.43**	0.9693	0.78
Soccer	31.24	0.8707	33.05	0.9113	**33.25**	0.9164	33.11	**0.9116**	4.66

SRHDT_f method and the proposed general model SRDT method on the testing sequence *City*. Fig. 4-11 shows that the aliasing region in the 16th frame of the testing sequence *City* has been better recovered by the data dependent model SRDT compared to the general model methods. As discussed in Section 4.5.1.2, the running time of the SRDT method is around 9 times faster than the SRHDT_f method. Thus, the data dependent model SRDT method can greatly reduce the prediction error while with much lower running complexity. However, the data dependent model may obtain lower PSNR compared with that of the general model. However, the sequence *Soccer* is an exceptional case. The main reason could be that there were too many movements of the players in the sequence to be captured by the 1st frame and 31st frame, such that the learned data dependent model cannot effectively super-resolve other frames.

4.6. CHAPTER SUMMARY

In this chapter, we have proposed a novel approach for efficient and effective single image super-resolution using decision tree and hierarchical decision trees. We suggested that a combination of classification process and regression process should be an efficient solution to the image super-resolution problem. The proposed image super-resolution

using decision tree method enables us to exploit fast image patch classification by comparing the intensity values of several pixel pairs, which has much lower complexity than solving sparse coding problem or looking for the cluster center using the *K*-means method. The linear regression model using simple regression feature keeps the computation cost of mapping low-resolution patch to its desired high-resolution patch low. To further boost the performance and improve the training efficiency of using single decision tree for image super-resolution, a hierarchical decision trees framework has been proposed. Each layer of the super-resolution decision tree in the hierarchical framework consistently refines the super-resolution results of the previous layer. The overall improvement of the hierarchical decision trees framework is over 0.4 dB in PSNR. As each training LR-HR patch pair can find its approximately rotated version, we have proposed to fuse the regression models from 4 relevant leaf nodes in a decision tree to form a more robust regression model for HR patch prediction which provides around 0.1 dB PSNR gain. Our extensive experimental results show that our proposed SRDT method achieves comparable SR quality as the sparse representation based method (the Peleg's method) and the deep learning based method (the SRCNN method) while more than 100 times faster. Furthermore, our proposed SRHDT_f method generates more than 0.3 dB higher PSNR image super-resolution results compared to the state-of-the-art fast method A+ with similar computational complexity.

CHAPTER 5 LEARNING HYBRID DCT-WIENER-BASED INTERPOLATION USING DUAL MMSE ESTIMATOR SCHEME

5.1. INTRODUCTION

In this chapter, we propose a hybrid DCT-Wiener-Based interpolation scheme using dual MMSE estimators to combine information from the input image and the training images. The rest of the organization of this chapter is as follow. Section 5.2 introduces a novel quantization table based DCT block classification method. Section 5.3 gives the proposed dual estimation scheme using both local and global estimators. Section 5.4 shows some experimental results and Section 5.5 draws a conclusion.

5.2. QUANTIZATION TABLE BASED DCT BLOCK CLASSIFICATION

It is obvious to point out that blocks with different characteristics should have different MMSE estimators, and the price of training a dedicated Wiener filter for each block could be very high. A compromise approach is to divide all blocks into a number of groups and assign each group an adaptive Minimum Mean Squared Error (MMSE) estimator. According to the characteristics of DCT coefficients, a novel quantization table (Qtable) based DCT block classification algorithm is proposed.

In the DCT domain, a $k \times k$ block has $k^2 - 1$ AC coefficients and the significant coefficients usually have large values. To alleviate the influences of small DCT coefficients, a Qtable is applied to quantize the DCT coefficients. The Qtable used in this chapter is the n times of the quantization table in JEPG standard, and n varies according to k (for example, when $k = 8$, $n = 5$, and when $k = 4$, $n = 2.5$). After the quantization, the

1079.8	-230.8	203.2	-138	59.5	-4.2	-25.5	20.4
14.2	-7.9	5	11.8	-16.4	16.5	-9.6	6.5
-0.4	0.2	-0.7	-0.3	0.5	-3.3	3.7	-0.5
3.8	-1.7	4.6	-1.4	1.9	-0.9	1.2	-0.9
1.3	-2.2	-1.8	2.1	-2	1.3	-1.7	2
1.8	-0.7	2.6	0.6	1.3	0.7	-0.1	-1
1	0	0.9	0.4	-0.2	0.7	1.4	0.7
1.4	-1	-3.7	0.8	1.8	-2.1	-0.1	-0.9

(a) The original 8×8 DCT block

80	55	50	80	120	200	255	305
60	60	70	95	130	290	300	275
70	65	80	120	200	285	345	280
70	85	110	145	255	435	400	310
90	110	185	280	340	545	515	385
120	175	275	320	405	520	565	460
245	320	320	435	515	605	600	505
360	460	460	490	560	500	515	495

(b) Sample quantization table

13	-4	4	-2	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(c) The DCT block after quantization

-4	4	0	0	0

(d) The 5 most significant AC coefficients after quantization

-1	1	0	0	0

(e) Final pattern for that DCT block

Fig. 5-1: The Qtable classification process.

pattern in DCT domain becomes clear and this will simplify the classification process, illustrated as an example in Fig. 5-1.

The 5 most significant AC coefficients (AC01, AC02, AC10, AC11, and AC20) are used for classification, and each coefficient will only retain the sign information. Each coefficient is represented using either 1, 0 or -1. There are totally 3^5=243 patterns after the Qtable classification, and thus 243 groups. Each group will have its own MMSE estimator. The reason for only using 5 most significant AC coefficients for classification is that using

more AC coefficients will make the number of estimators too large, which will require an extremely large training data set.

This proposed Qtable based DCT block classification method provides a simple DCT blocks clustering algorithm and can easily connect the global and the local estimators as presented in the following sections.

5.3. HYBRID ESTIMATION SCHEME USING LOCAL AND GLOBAL ESTIMATORS

A dual estimator scheme which chooses either a local or global MMSE estimator is suggested in this section. The local MMSE estimator makes use of the input image only, whereas the global MMSE estimator relies on pre-training of external images. The local MMSE estimator is learnt from the image itself and features with adaptive up-sampling results. The global MMSE estimator is used when there are not sufficient training data to form a local MMSE estimator. The global MMSE estimators (one for each pattern in Section 5.2) constructed off line making use of a set of training images, which provide a stable but non-adaptive performance to all images.

In the previous works [68, 72, 73], they use patches from low-resolution image to predict the patches in high-resolution image. While in our approach, we first up-sample the low-resolution image to the same size as the high-resolution image using spatial interpolation method and then use the patches from the spatially up-sampled image to predict the patches in the high-resolution image. This modification is to maximize the correlation between the source patch and the desired patch.

5.3.1. GLOBAL MMSE ESTIMATORS TRAINING

A set of 243 global MMSE estimators are offline trained using 12 images (Kodim 1~12), as shown in Fig. 5-2. The original high-resolution training image is denoted as

Fig. 5-2. The 12 training images (Kodim 1~12).

HR_0^T ($2m \times 2n$), and its DCT domain down-sampled image (using the dyadic down-sample scheme in [62]) and then 6-tap Wiener filter [78] up-sampled image is denoted as HR_w^T ($2m \times 2n$). HR_0^T and HR_w^T are partitioned into overlapping $k \times k$ blocks. When a block from HR_w^T falls into the ith group using the Qtable classification, the block pairs from HR_0^T and HR_w^T are vectorized and grouped into matrices $\mathbf{X}^{\mathbf{i}}_{\mathbf{H}_0^T}$ ($k^2 \times N_{\max}$) and $\mathbf{X}^{\mathbf{i}}_{\mathbf{H}_w^T}$ ($k^2 \times N_{\max}$) respectively, where N_{max} is the maximum number of training data in each group. The global Wiener filter coefficients of the ith group are given by equation (5-1).

$$\mathbf{H}^{\mathbf{i}}_{\mathbf{global}} = E[\mathbf{X}^{\mathbf{i}}_{\mathbf{H}_0^T} \mathbf{X}^{\mathbf{i}\ \mathbf{T}}_{\mathbf{H}_w^T}] E[\mathbf{X}^{\mathbf{i}}_{\mathbf{H}_w^T} \mathbf{X}^{\mathbf{i}\ \mathbf{T}}_{\mathbf{H}_w^T}]^{-1}, \tag{5-1}$$

where $\mathbf{H}^{\mathbf{i}}_{\mathbf{global}}$ is the Wiener filter coefficient matrix for the ith group of the global MMSE estimators, which has the size of $k^2 \times k^2$ and $i \in [1,243]$.

5.3.2. LOCAL MMSE ESTIMATORS TRAINING

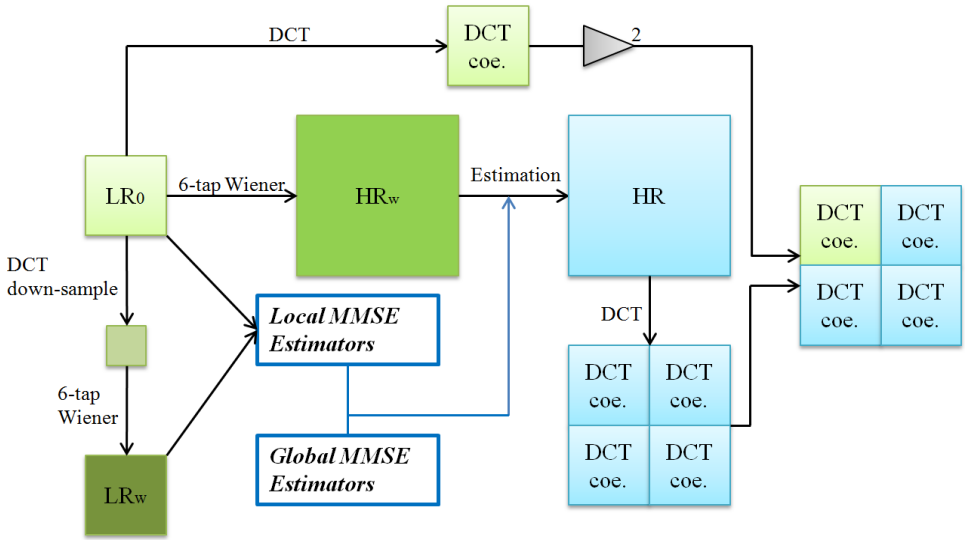


Fig. 5-3: The proposed interpolation scheme.

The input low-resolution image LR_0 and its DCT down-sampled and then 6-tap Wiener filter up-sampled image LR_w are partitioned into overlapping $k \times k$ blocks. The blocks pairs from LR_0 and LR_w are vectorized and grouped into matrices $\mathbf{X}^i_{L_0}$ $(k^2 \times N_i)$ and $\mathbf{X}^i_{L_w}$ $(k^2 \times N_i)$ respectively, if the block from LR_w falls into the i[th] group using the Qtable classification method, where N_i is the number of training data in the i[th] group. The local Wiener filter coefficient of the i[th] group is given by Eq. (5-2).

$$\mathbf{H}^i_{\text{local}} = E[\mathbf{X}^i_{L_0} \mathbf{X}^{i\ T}_{L_w}] E[\mathbf{X}^i_{L_w} \mathbf{X}^{i\ T}_{L_w}]^{-1}, \tag{5-2}$$

where $\mathbf{H}^i_{\text{local}}$ is the Wiener filter coefficient matrix for the i[th] group of the local estimators, which has the size of $k^2 \times k^2$ and $i \in [1,243]$.

5.3.3 THE HYBRID ESTIMATION FRAMEWORK

Let us clarify when two types of estimators are used. Owing to the size limitation of the low-resolution image, only parts of the local estimators will have enough training data (more than 200). If $\mathbf{H}^{i}_{local}$ has enough training data, $\mathbf{H}^{i}_{local}$ will replace $\mathbf{H}^{i}_{local}$ to up-sample the low-resolution image as stated in Eq. (5-3).

$$\mathbf{H}^{k} = \begin{cases} \mathbf{H}^{k}_{local} & \text{if } N_{k} \geq 200 \\ \mathbf{H}^{k}_{global} & \text{otherwise} \end{cases} \tag{5-3}$$

where the threshold 200 is determined by the cross-validation.

After finishing the local estimators online training, LR_0 (m×n) is up-sampled using 6-tap Wiener filter [78]. The resultant image HR_w (2m×2n) is partitioned into k×k overlapping blocks. Blocks $\mathbf{X}$ (k×k) from HR_w are transformed to DCT domain, and classified into one of the 243 groups using the Qtable classification method as discussed in **Section 5.2**. $\mathbf{X}$ is then transformed into column vector $\mathbf{X}_{col}$ ($k^2 \times 1$). The up-sampled high-resolution image block $\mathbf{Y}$ (k×k) is reshaped using the estimated high-resolution column vector $\mathbf{Y}_{col}$ ($k^2 \times 1$) by Eq. (5-3) and Eq. (5-4).

$$\mathbf{Y}_{col} = \mathbf{H}^{i}\mathbf{X}_{col}. \tag{5-4}$$

Overlapping blocks are applied in this scheme, so each pixel has multiple estimations and the final pixel value is the average of all estimations. After the high-resolution image *HR* is estimated using the dual estimators, a hybrid DCT-Wiener-Based process is applied to *HR* using the 4×4 low-frequency components from LR_0 to replace those in *HR* for non-overlapping 8×8 DCT blocks.

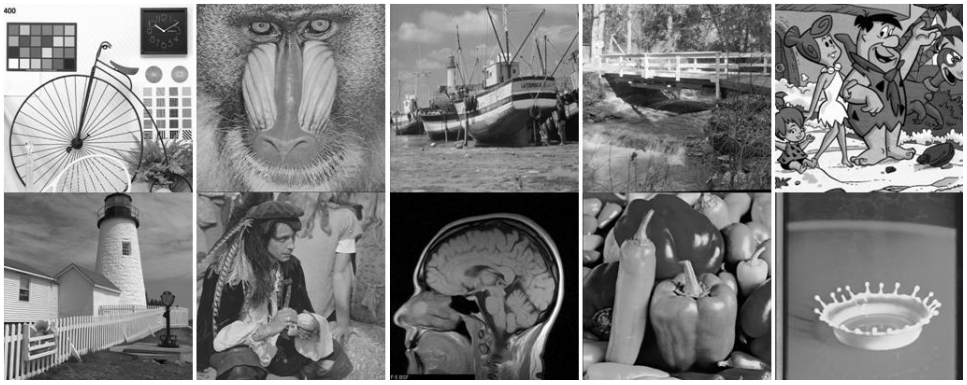


Fig. 5-4: The 10 testing images.

5.4. EXPERIMENTAL RESULTS

To evaluate the proposed approach, 10 images (512×512) with various contents have been selected for testing.

Table 5-1 shows the PSNR (dB) comparison with different algorithms [62, 71-73] for two-times up-sampling. The computation complexity is the amount of computation time compared with zero-padding algorithm [62]. In Table 5-1, when only global estimators (k=8) are used for up sampling, around 0.7dB and 0.3dB improvements are achieved compared with original hybrid scheme [71] and hybrid scheme with learnt Wiener filter [72] respectively. The results confirm the idea that using different MMSE estimator for a patch with different characteristics will improve the prediction accuracy. When both local and global estimators are correctly selected in up-sampling, the proposed dual estimator scheme (k=8) provides 0.8dB gain against the original hybrid scheme [71] and 0.4dB gain against the hybrid scheme using learnt Wiener filter [72]. The reason is that the original low-resolution image and the original high-resolution image contain some similar but particular feature, and the learnt local estimator can help to recover these relevant features more effectively than global estimators. On a further investigation, we also propose to use

TABLE 5-1
THE PSNR (DB) OF THE TESTING IMAGES COMPARED WITH DIFFERENT ALGORITHMS.

Images	**Zero-padding [13]**	**Hybrid scheme [19]**	**Hybrid Learnt Wiener [22]**	**Adaptive *k*-NN MMSE [23]**	**Only global estimators (*k*=8)**	**Proposed dual estimation scheme (*k*=8)**	**Proposed dual estimation scheme (*k*=4)**
Bicycle	21.714	21.602	21.788	**23.082**	22.019	22.535	**22.849**
Baboon	24.799	24.755	24.912	**25.117**	**25.020**	24.960	25.012
Boat	30.415	30.429	30.876	**31.635**	31.111	31.083	**31.259**
Bridge	26.981	26.921	27.251	**27.555**	27.332	27.349	**27.433**
Flin	27.458	27.924	28.515	29.664	29.128	29.659	**29.983**
Lighthous	26.831	26.877	27.283	**28.198**	27.395	27.026	**27.471**
Man	31.539	31.650	32.017	**32.757**	32.340	32.381	**32.578**
Mri	35.242	35.470	35.605	35.754	35.622	35.594	**35.779**
Pepper	32.779	32.339	33.001	**34.242**	33.730	**33.963**	33.770
Splash	39.780	40.023	40.399	**43.297**	40.792	41.150	**41.735**
Average	29.754	29.799	30.165	**31.130**	30.449	30.570	**30.787**
Improve	1.033	0.988	0.622	-0.343	0.338	0.217	-
Complexi	1	1.8	2.4	683	15	24	46

dual estimator scheme (*k*=4), which offers an addition of 0.217 dB gain as compared to the proposed algorithm for *k*=8. In the literature review, the zero-padding based approaches [61, 62, 64-67, 69] used larger and larger DCT block for zero-padding and the PSNR performance of which improves. The main reason is due to the de-correlation property and energy compaction property of the Discrete Cosine transform. However, a smaller DCT block may have stronger correlation existing among the coefficients and there is a larger chance to have a simple pattern within the block. For the prediction-based approach in the spatial domain, using smaller block may be a better choice.

Figure 5-5 shows the subjective comparison among different algorithms. The proposed algorithm can help to recover edges more faithfully with less ringing effect and overshoots and give more pleasant appearance compared with the original hybrid scheme [71] and the hybrid scheme using learnt Wiener filter [72].

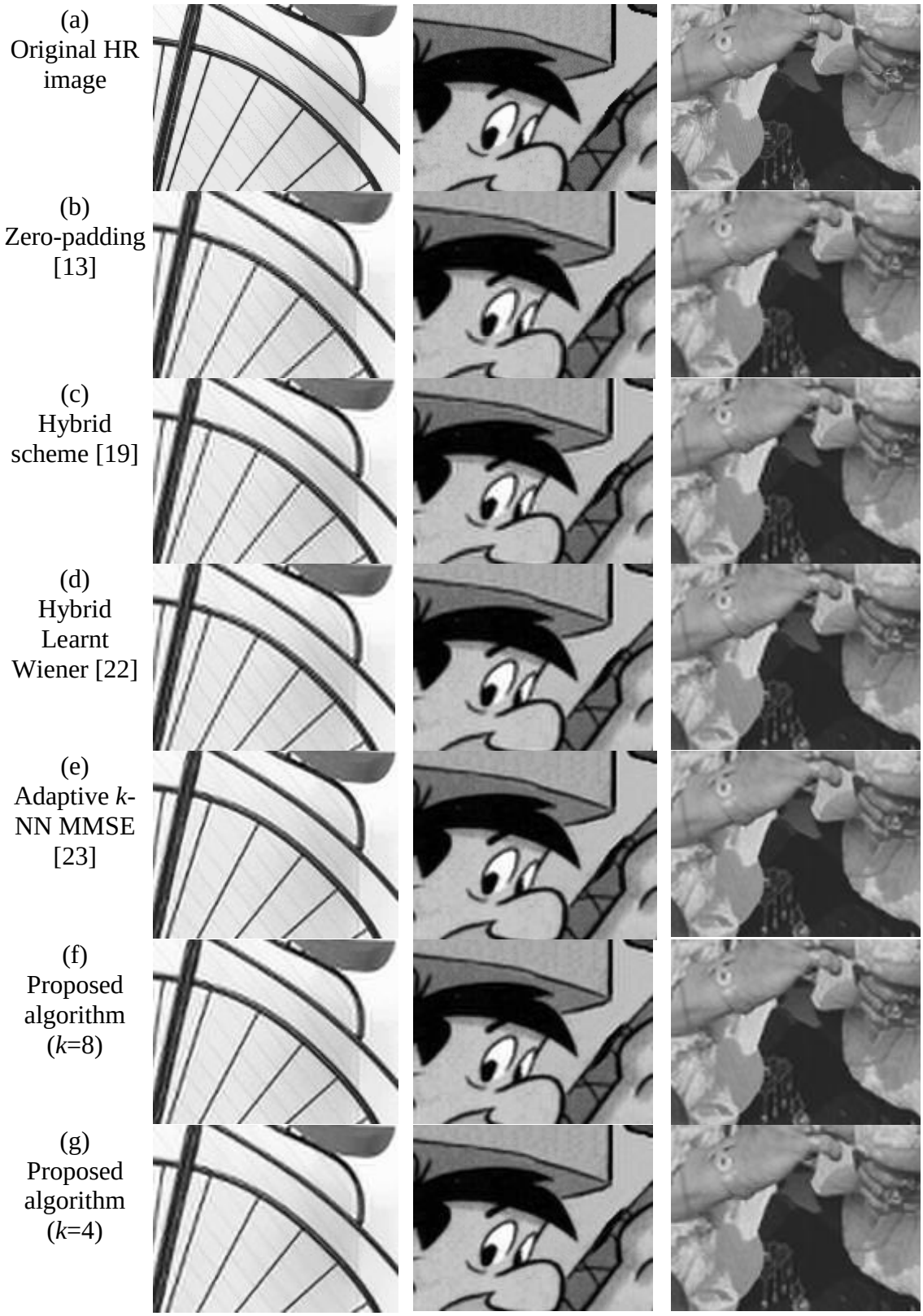


Fig. 5-5. Subjective comparison of different algorithms

The adaptive *k*-NN MMSE algorithm [73] gives the best PSNR performance; however, the average processing time is much longer (20 times longer) than the proposed algorithm. The proposed algorithm is more efficient if the quality-speed trade-off is considered. Furthermore, the subjective quality of the proposed algorithm is very close to that of adaptive *k*-NN MMSE estimation [73].

5.5. CHAPTER SUMMARY

An hybrid DCT-Wiener-Based interpolation scheme using dual MMSE estimator scheme is proposed. A quantization table based block classification algorithm is applied on the DCT domain, and all DCT blocks are classified into a number of groups. Each group of DCT block has its own estimator. A set of local estimators are trained using the original low-resolution image and its DCT down-sampled then spatially up-sampled image. In order to resolve the problem of inefficient of number of on-line samples, a set of global estimators are pre-trained. Both local and global estimators are used in image up-sampling according to the number of training data available to form a local estimator. Results of our experimental work show that the proposed dual MMSE estimator scheme is a fast approach which provides around 1dB PSNR improvement compared with the original hybrid scheme and offers more pleasant and natural visual quality.

CHAPTER 6 CONCLUSIONS AND FUTURE WORK

6.1. CONCLUSIONS

In this thesis, we have focused on solving image interpolation and image super-resolution problem performed on both spatial domain and transform domain. The ill-posed inverse image up-sampling problem is settled in a classification based manner.

In Chapter 3, we have proposed a two-stage random forests scheme for image interpolation. The proposed fast interpolation via random forests (FIRF) method achieves the state-of-the-art image interpolation quality while is about 70 to 320 times faster over the sparse representation model based method NARM [58]. The fast testing speed of the proposed FIFR method is achieved through a combination of simple binary tests on non-leaf nodes and simple linear regression models on leaf nodes. The proposed constraint on binary split ensures that good binary splits are selected and compact decision trees are learned during training which facilitates fast testing speed and high image interpolation quality. The proposed two-stage random forests scheme further improves image interpolation performance by around 0.2 dB in PSNR. By varying the number of decision trees within a stage and the number of stages applied, the proposed FIRF method can provide scalable image interpolation which can be adaptive to devices with different computational capabilities ability.

In Chapter 4, decision tree is comprehensively studied for effective and efficient image super-resolution. We have found that decision tree is more suitable than random forests (an ensemble of decision trees) for efficient image up-sampling. Although random forests indeed has better generalization ability, around 0.1 dB improvement in PSNR is achieved

under the cost of 3-5 times testing time. The proposed image super-resolution using decision tree (SRDT) method achieves 100 times faster image super-resolution speed when it provides comparable image super-resolution quality as the sparse representation based method (the Peleg's method [35]) and the deep learning based method (the SRCNN method [34]). The hierarchical decision trees framework (SRHDT) further improves the image super-resolution quality by a large margin with relatively short computational load increase. The overall PSNR improvement is more than 0.4 dB over the SRDT method. We have proposed further to fuse regression models on relevant leaf nodes within a single decision tree to pursuit even better image super-resolution quality. The proposed hierarchical decision trees with fused regression models (SRHDT_f) method provides another 0.1 dB increase in PSNR and achieves 0.3~0.4 dB improvement over the state-of-the-art A+ method [36]. A data dependent model also has been proposed to improve the super-resolution results for video. In conclusion, we have proposed to quest fast and high quality image super-resolution algorithms through single decision tree and hierarchical decision trees, and with a fused regression models scheme.

In Chapter 5, a dual MMSE scheme is proposed to improve the Hybrid-DCT-Wiener-based interpolation method [71] by making use of the useful information from the input low-resolution image and the training images. We have proposed a quantization table based algorithm to simplify the coefficients on a DCT block and classify the DCT blocks into a set of defined groups. Based on the number of training data from the input low-resolution image within each group, the pre-trained MMSE estimators learning from the training images is decided whether to be selected for up-sampling. The proposed dual

MMSE scheme offers around 1 dB gain in PSNR over the original Hybrid-Wiener-based method [71].

6.2. FUTURE RESEARCH DIRECTIONS

There are a lot of possible research directions aroused by the research works studied in this thesis.

Most image up-sampling algorithms apply pre-trained dictionaries or learn models from the input LR images for image upscaling. The future research direction of image up-sampling may be how to effectively combine information from both internal examples and external learned dictionaries. The internal examples is an important prior knowledge for image up-sampling. They could help to recover high ambiguity regions which are difficult for external based image up-sampling algorithms to deal with. Or, the information from input LR image could make the learned dictionaries adaptive to varies blur conditions.

For very large scale image up-sampling (more than 8 times), the contemporary results seem distorted, unnatural and without rich texture. The research could try to make the upscaled images look natural rather than be exactly the same as the ground-truth image.

For classification based image up-sampling methods, the training samples in each class have very similar appearances which suggest an underling highly structured relationship. The number of coefficients of the regression models could be reduced to make the algorithm more efficient. Some preliminary experimental results show that the regression matrix could be reduced from 36×36 to 1×36 for some classes without significant quality loss.

REFERENCES

[1] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael, "Learning low-level vision," *International journal of computer vision,* vol. 40, pp. 25-47, 2000.

[2] W. T. Freeman, T. R. Jones, and E. C. Pasztor, "Example-based super-resolution," *Computer Graphics and Applications, IEEE,* vol. 22, pp. 56-65, 2002.

[3] H. Chang, D.-Y. Yeung, and Y. Xiong, "Super-resolution through neighbor embedding," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on,* 2004, pp. I-I.

[4] W. Fan and D.-Y. Yeung, "Image hallucination using neighbor embedding over visual primitive manifolds," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on,* 2007, pp. 1-7.

[5] K. S. Ni and T. Q. Nguyen, "Image superresolution using support vector regression," *Image Processing, IEEE Transactions on,* vol. 16, pp. 1596-1610, 2007.

[6] X. Li, K. M. Lam, G. Qiu, L. Shen, and S. Wang, "An efficient example-based approach for image super-resolution," in *Neural Networks and Signal Processing, 2008 International Conference on,* 2008, pp. 575-580.

[7] A. Marquina and S. J. Osher, "Image super-resolution by TV-regularization and Bregman iteration," *Journal of Scientific Computing,* vol. 37, pp. 367-382, 2008.

[8] T.-M. Chan, J. Zhang, J. Pu, and H. Huang, "Neighbor embedding based super-resolution algorithm through edge detection and feature selection," *Pattern Recognition Letters,* vol. 30, pp. 494-502, 2009.

[9] S. Dai, M. Han, W. Xu, Y. Wu, Y. Gong, and A. K. Katsaggelos, "Softcuts: a soft edge smoothness prior for color image super-resolution," *Image Processing, IEEE Transactions on,* vol. 18, pp. 969-981, 2009.

[10] D. Glasner, S. Bagon, and M. Irani, "Super-resolution from a single image," in *Computer Vision, 2009 IEEE 12th International Conference on*, 2009, pp. 349-356.

[11] M. Protter, M. Elad, H. Takeda, and P. Milanfar, "Generalizing the nonlocal-means to super-resolution reconstruction," *Image Processing, IEEE Transactions on,* vol. 18, pp. 36-51, 2009.

[12] K. I. Kim and Y. Kwon, "Single-image super-resolution using sparse regression and natural image prior," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 32, pp. 1127-1133, 2010.

[13] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *Image Processing, IEEE Transactions on,* vol. 19, pp. 2861-2873, 2010.

[14] H. Zhang, J. Yang, Y. Zhang, and T. S. Huang, "Non-local kernel regression for image and video restoration," in *Computer Vision–ECCV 2010*, ed: Springer, 2010, pp. 566-579.

[15] W. Dong, L. Zhang, G. Shi, and X. Wu, "Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization," *Image Processing, IEEE Transactions on,* vol. 20, pp. 1838-1857, 2011.

[16] G. Freedman and R. Fattal, "Image and video upscaling from local self-examples," *ACM Transactions on Graphics (TOG),* vol. 30, p. 12, 2011.

[17] A. Giachetti and N. Asuni, "Real-time artifact-free image upscaling," *Image Processing, IEEE Transactions on*, vol. 20, pp. 2760-2768, 2011.

[18] H. He and W.-C. Siu, "Single image super-resolution using Gaussian process regression," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 2011, pp. 449-456.

[19] J. Sun, Z. Xu, and H.-Y. Shum, "Gradient profile prior and its applications in image super-resolution and enhancement," *Image Processing, IEEE Transactions on*, vol. 20, pp. 1529-1542, 2011.

[20] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," 2012.

[21] X. Gao, K. Zhang, D. Tao, and X. Li, "Image super-resolution with sparse neighbor embedding," *Image Processing, IEEE Transactions on*, vol. 21, pp. 3194-3205, 2012.

[22] S. Gu, N. Sang, and F. Ma, "Fast image super resolution via local regression," in *Pattern Recognition (ICPR), 2012 21st International Conference on*, 2012, pp. 3128-3131.

[23] K.-W. Hung and W.-C. Siu, "Single image super-resolution using iterative Wiener filter," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, 2012, pp. 1269-1272.

[24] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Curves and Surfaces*, ed: Springer, 2012, pp. 711-730.

[25] K. Zhang, X. Gao, D. Tao, and X. Li, "Single image super-resolution with non-local means and steering kernel regression," *Image Processing, IEEE Transactions on,* vol. 21, pp. 4544-4556, 2012.

[26] W. Dong, L. Zhang, G. Shi, and X. Li, "Nonlocally centralized sparse representation for image restoration," *Image Processing, IEEE Transactions on,* vol. 22, pp. 1620-1630, 2013.

[27] L. He, H. Qi, and R. Zaretzki, "Beta process joint dictionary learning for coupled feature spaces with application to single image super-resolution," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, 2013, pp. 345-352.

[28] R. Timofte, V. De, and L. Van Gool, "Anchored neighborhood regression for fast example-based super-resolution," in *Computer Vision (ICCV), 2013 IEEE International Conference on*, 2013, pp. 1920-1927.

[29] L. Wang, S. Xiang, G. Meng, H. Wu, and C. Pan, "Edge-directed single-image super-resolution via adaptive gradient magnitude self-interpolation," *Circuits and Systems for Video Technology, IEEE Transactions on,* vol. 23, pp. 1289-1299, 2013.

[30] H. Xu, G. Zhai, and X. Yang, "Single image super-resolution with detail enhancement based on local fractal analysis of gradient," *Circuits and Systems for Video Technology, IEEE Transactions on,* vol. 23, pp. 1740-1754, 2013.

[31] C.-Y. Yang and M.-H. Yang, "Fast direct super-resolution by simple functions," in *Computer Vision (ICCV), 2013 IEEE International Conference on*, 2013, pp. 561-568.

[32] J. Yang, Z. Lin, and S. Cohen, "Fast image super-resolution based on in-place example regression," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, 2013, pp. 1059-1066.

[33] Z. Cui, H. Chang, S. Shan, B. Zhong, and X. Chen, "Deep network cascade for image super-resolution," in *Computer Vision–ECCV 2014*, ed: Springer, 2014, pp. 49-64.

[34] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Computer Vision–ECCV 2014*, ed: Springer, 2014, pp. 184-199.

[35] T. Peleg and M. Elad, "A statistical prediction model based on sparse representations for single image super-resolution," *Image Processing, IEEE Transactions on,* vol. 23, pp. 2569-2582, 2014.

[36] R. Timofte, V. De Smet, and L. Van Gool, "A+: Adjusted anchored neighborhood regression for fast super-resolution," in *Computer Vision--ACCV 2014*, ed: Springer, 2014, pp. 111-126.

[37] J. Huang, W. Siu, and T. Liu, "Fast Image Interpolation via Random Forests," *Image Processing, IEEE Transactions on,* vol. 24, pp. 3232-3245, 2015.

[38] K. Zhang, D. Tao, X. Gao, X. Li, and Z. Xiong, "Learning Multiple Linear Mappings for Efficient Single Image Super-Resolution," *Image Processing, IEEE Transactions on,* vol. 24, pp. 846-861, 2015.

[39] H. S. Hou and H. Andrews, "Cubic splines for image interpolation and digital filtering," *Acoustics, Speech and Signal Processing, IEEE Transactions on,* vol. 26, pp. 508-517, 1978.

[40] R. G. Keys, "Cubic convolution interpolation for digital image processing," *Acoustics, Speech and Signal Processing, IEEE Transactions on,* vol. 29, pp. 1153-1160, 1981.

[41] M. Irani and S. Peleg, "Motion analysis for image enhancement: Resolution, occlusion, and transparency," *Journal of Visual Communication and Image Representation,* vol. 4, pp. 324-335, 1993.

[42] K. Jensen and D. Anastassiou, "Subpixel edge localization and the interpolation of still images," *Image Processing, IEEE Transactions on,* vol. 4, pp. 285-295, 1995.

[43] T. M. Lehmann, C. Gönner, and K. Spitzer, "Addendum: B-spline interpolation in medical image processing," *Medical Imaging, IEEE Transactions on,* vol. 20, pp. 660-665, 2001.

[44] X. Li and M. T. Orchard, "New edge-directed interpolation," *Image Processing, IEEE Transactions on,* vol. 10, pp. 1521-1527, 2001.

[45] H. Shi and R. Ward, "Canny edge based image expansion," in *Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on*, 2002, pp. I-785-I-788 vol. 1.

[46] T. Blu, P. Thévenaz, and M. Unser, "Linear interpolation revitalized," *Image Processing, IEEE Transactions on,* vol. 13, pp. 710-719, 2004.

[47] D. D. Muresan, "Fast edge directed polynomial interpolation," in *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, 2005, pp. II-990-3.

[48] L. Zhang and X. Wu, "An edge-guided image interpolation algorithm via directional filtering and data fusion," *Image Processing, IEEE Transactions on,* vol. 15, pp. 2226-2238, 2006.

[49] Q. Wang and R. K. Ward, "A new orientation-adaptive interpolation method," *Image Processing, IEEE Transactions on,* vol. 16, pp. 889-900, 2007.

[50] X. Zhang and X. Wu, "Image interpolation by adaptive 2-D autoregressive modeling and soft-decision estimation," *Image Processing, IEEE Transactions on,* vol. 17, pp. 887-896, 2008.

[51] K. S. Ni and T. Q. Nguyen, "An adaptable-nearest neighbors algorithm for MMSE image interpolation," *Image Processing, IEEE Transactions on,* vol. 18, pp. 1976-1987, 2009.

[52] W.-S. Tam, C.-W. Kok, and W.-C. Siu, "Modified edge-directed interpolation for images," *Journal of Electronic imaging,* vol. 19, pp. 013011-013011-20, 2010.

[53] C.-S. Wong and W.-C. Siu, "Further improved edge-directed interpolation and fast EDI for SDTV to HDTV conversion," in *Proc. of European Signal Processing Conference*, 2010, pp. 23-27.

[54] C.-S. Wong and W.-C. Siu, "Adaptive directional window selection for edge-directed Interpolation," in *Computer Communications and Networks (ICCCN), 2010 Proceedings of 19th International Conference on*, 2010, pp. 1-6.

[55] X. Liu, D. Zhao, R. Xiong, S. Ma, W. Gao, and H. Sun, "Image interpolation via regularized local linear regression," *Image Processing, IEEE Transactions on,* vol. 20, pp. 3455-3469, 2011.

[56] K.-W. Hung and W.-C. Siu, "Fast image interpolation using the bilateral filter," *Image Processing, IET,* vol. 6, pp. 877-890, 2012.

[57] K.-W. Hung and W.-C. Siu, "Robust soft-decision interpolation using weighted least squares," *Image Processing, IEEE Transactions on,* vol. 21, pp. 1061-1069, 2012.

[58] W. Dong, L. Zhang, R. Lukac, and G. Shi, "Sparse representation based image interpolation with nonlocal autoregressive modeling," *Image Processing, IEEE Transactions on,* vol. 22, pp. 1382-1394, 2013.

[59] H. Kirshner, A. Bourquard, J. P. Ward, M. Porat, and M. Unser, "Adaptive Image Resizing Based on Continuous-Domain Stochastic Modeling," *Image Processing, IEEE Transactions on*, vol. 23, pp. 413-423, 2014.

[60] Y. Romano, M. Protter, and M. Elad, "Single image interpolation via adaptive nonlocal sparsity-based modeling," *Image Processing, IEEE Transactions on,* vol. 23, pp. 3085-3098, 2014.

[61] J. Agbinya, "Interpolation using the discrete cosine transform," *Electronics letters,* vol. 28, pp. 1927-1928, 1992.

[62] R. Dugad and N. Ahuja, "A fast scheme for image size change in the compressed domain," *Circuits and Systems for Video Technology, IEEE Transactions on,* vol. 11, pp. 461-474, 2001.

[63] J. Jiang and G. Feng, "The spatial relationship of DCT coefficients between a block and its sub-blocks," *Signal Processing, IEEE Transactions on,* vol. 50, pp. 1160-1169, 2002.

[64] J. Mukherjee and S. K. Mitra, "Image resizing in the compressed domain using subband DCT," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 12, pp. 620-627, 2002.

[65] H. Park, Y. Park, and S.-K. Oh, "L/M-fold image resizing in block-DCT domain using symmetric convolution," *Image Processing, IEEE Transactions on,* vol. 12, pp. 1016-1034, 2003.

[66] Y. Park and H. Park, "Design and analysis of an image resizing filter in the block-DCT domain," *Circuits and Systems for Video Technology, IEEE Transactions on,* vol. 14, pp. 274-279, 2004.

[67] J. Mukherjee and S. K. Mitra, "Arbitrary resizing of images in DCT space," *IEE Proceedings-Vision, Image and Signal Processing,* vol. 152, pp. 155-164, 2005.

[68] M.-K. Cho and B.-U. Lee, "Discrete cosine transform domain image resizing using correlation of discrete cosine transform coefficients," *Journal of Electronic Imaging,* vol. 15, pp. 033009-033009-6, 2006.

[69] V. Patil, R. Kumar, and J. Mukherjee, "A fast arbitrary factor video resizing algorithm," *Circuits and Systems for Video Technology, IEEE Transactions on,* vol. 16, pp. 1164-1171, 2006.

[70] S.-J. Park and J. Jeong, "Hybrid image upsampling method in the discrete cosine transform domain," *Consumer Electronics, IEEE Transactions on,* vol. 56, pp. 2615-2622, 2010.

[71] Z. Wu, H. Yu, and C. W. Chen, "A new hybrid DCT-Wiener-based interpolation scheme for video intra frame up-sampling," *Signal Processing Letters, IEEE,* vol. 17, pp. 827-830, 2010.

[72] K.-W. Hung and W.-C. Siu, "Hybrid DCT-Wiener-based interpolation via learnt Wiener filter," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013, pp. 1419-1423.

[73] K.-W. Hung and W.-C. Siu, "Novel DCT-Based Image Up-Sampling Using Learning-Based Adaptive-NN MMSE Estimation," *Circuits and Systems for Video Technology, IEEE Transactions on,* vol. 24, pp. 2018-2033, 2014.

[74] Q. Wang, X. Tang, and H. Shum, "Patch based blind image super resolution," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, 2005, pp. 709-716.

[75] N. Efrat, D. Glasner, A. Apartsin, B. Nadler, and A. Levin, "Accurate blur models vs. image priors in single image super-resolution," in *Computer Vision (ICCV), 2013 IEEE International Conference on*, 2013, pp. 2832-2839.

[76] T. Michaeli and M. Irani, "Nonparametric blind super-resolution," in *Computer Vision (ICCV), 2013 IEEE International Conference on*, 2013, pp. 945-952.

[77] T. Michaeli and M. Irani, "Blind deblurring using internal patch recurrence," in *Computer Vision–ECCV 2014*, ed: Springer, 2014, pp. 783-798.

[78] "Advanced Video Coding for Generic Audiovisual Services ITU-T and ISO/IEC JTC1," *Rec. H.264-ISO/IEC 14496-10 AVC*, 2003.

[79] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*: MIT press, 2012.

[80] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, pp. 273-297, 1995.

[81] L. Breiman, "Arcing classifier (with discussion and a rejoinder by the author)," *The annals of statistics*, vol. 26, pp. 801-849, 1998.

[82] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*: CRC press, 1984.

[83] B. Yao, A. Khosla, and L. Fei-Fei, "Combining randomization and discrimination for fine-grained image categorization," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 2011, pp. 1577-1584.

[84] S. Rota Bulo and P. Kontschieder, "Neural decision forests for semantic image labelling," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 2014, pp. 81-88.

[85] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, pp. 3-42, 2006.

[86] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5-32, 2001.

[87] V. Lepetit and P. Fua, "Keypoint recognition using randomized trees," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, pp. 1465-1479, 2006.

[88] A. Bosch, A. Zisserman, and X. Munoz, "Image classification using random forests and ferns," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, 2007, pp. 1-8.

[89] J. Shotton, M. Johnson, and R. Cipolla, "Semantic texton forests for image categorization and segmentation," in *Computer vision and pattern recognition, 2008. CVPR 2008. IEEE Conference on*, 2008, pp. 1-8.

[90] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky, "Hough forests for object detection, tracking, and action recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, pp. 2188-2202, 2011.

[91] A. Criminisi, J. Shotton, and E. Konukoglu, *Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning*: Now, 2012.

[92] M. Dantone, J. Gall, G. Fanelli, and L. Van Gool, "Real-time facial feature detection using conditional regression forests," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 2012, pp. 2578-2585.

[93] P. Dollár and C. L. Zitnick, "Structured forests for fast edge detection," in *Computer Vision (ICCV), 2013 IEEE International Conference on*, 2013, pp. 1841-1848.

[94] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *Image Processing, IEEE Transactions on,* vol. 13, pp. 600-612, 2004.

[95] L. Zhang, L. Zhang, X. Mou, and D. Zhang, "FSIM: a feature similarity index for image quality assessment," *Image Processing, IEEE Transactions on*, vol. 20, pp. 2378-2386, 2011.

[96] J.-J. Huang and W.-C. Siu, "Practical application of random forests for super-resolution imaging," in *Circuits and Systems (ISCAS), 2015 IEEE International Symposium on*, 2015, pp. 2161-2164.