SONA COLLEGE OF TECHNOLOGY

Learning is a Celebration!

| An Autonomous Institution |

Department of Electronics and Communication Engineering

TRANSFORMS AND ALGORITHM IN SIGNAL AND IMAGE PROCESSING

2017-18

Junction Main Road
Suramangalam (PO)
Salem-636 005. TN. India
www.sonatech.ac.in

EDITOR'S BOARD

Editorial Head

Dr.R.S.Sabeenian,
Professor &Head, Dept of ECE, Head R&D Sona SIPRO.

Editorial members:

1. Ms.M.Jamuna Rani,
   Assist.Professor(Sr.G)

2. Dr.S.Vijayalakshmi,
   Assistant Professor

3. Mr.R.Anand,
   Assistant Professor/ECE

Magazine Coordinator

M.Senthil Vadivu,
Assistant Professor/ECE

Student Members:

1. D.Karthik

2. Ajith Kumar

Preface

The field of signal and image processing encompasses the theory and practice of algorithms and hardware that convert signals produced by artificial or natural means into a form useful for a specific purpose. The signals might be speech, audio, images, video, sensor data, telemetry, electrocardiograms, or seismic data, among others; possible purposes include transmission, display, storage, interpretation, classification, segmentation, or diagnosis.

Current research in digital signal processing includes robust and low complexity filter design, signal reconstruction, filter bank theory, and wavelets. In statistical signal processing, the areas of research include adaptive filtering, learning algorithms for neural networks, spectrum estimation and modeling, and sensor array processing with applications in sonar and radar. Image processing work is in restoration, compression, quality evaluation, computer vision, and medical imaging. Speech processing research includes modeling, compression, and recognition. Video compression, analysis, and processing projects include error concealment technique for 3D compressed video, automated and distributed crowd analytics, stereo-to-auto stereoscopic 3D video conversion, virtual and augmented reality.

TABLE OF CONTENTS

HARDWARE DEVELOPMENTS FOR IMAGE PROCESSING AND IT'S APPLICATIONS ON CRYPTOGRAPHY

Image thresholding as segmentation:

The first stage that we can think of in all stage of image processing and analysis is image binarization (i.e. to make binary image, the image should contain any two-pixel values either 0 or 1 in contrast with gray images which can contains 255 pixel values for 8 bit image) which poses as one of the serious problem in applications like machine vision, pattern recognition, target tracking and image segmentation where the gray level information is required to reduce to bi-level information. In order to extract the useful information from an image it needs to be divided into distinct components like foreground (where pixel value is '1') and background (where pixel value is '0') objects for further analysis where most often the gray level pixels of foreground components are quite different from that of background and in this context a very crucial and significant technique available in literature known as thresholding is applied which is the process of partitioning pixels in the images into object and background classes based upon the relationship between the gray level value of a pixel and the significant parameter threshold to separate the object from the background, finding the correct value of which to separate an image into desirable foreground and background remains a very crucial step in image processing domain. Because of its efficient performance and simplicity in theory, thresholding techniques have been studied extensively and many thresholding methods have been published so far A dedicated custom hardware on FPGA can process image in real time with fairly lower processing cost and power compare to software. Field Programmable Gate Arrays (FPGAs), can be used to speed up image processing applications. An application implemented on an FPGA can be one to two orders of magnitude faster than the same application implemented in software where parallel computation of hardware should be one of the important merit of hardware platform. In this paper we have designed and implemented an adaptive thresholding as a function of the image pixel intensities. Finding an optimal threshold value leading to an effective binarized image requires skill as the choice of the method must be done judiciously. After an initial pre-processing of the image the thresholding has been applied where the threshold value is dependent on the nature of the image which becomes a very dominant factor at the end.

Secured transmission of the image data between multiple FPGA platforms

Image applications are used in internet, multimedia systems, medical, and telemedicine, military for communication whereas this communication is not secured at all and could fall in the prey of any attacker who could attack on the transmitted images thus secretes and sensitiveness of the images can be disclosed to the unauthorized person. Thus, in this paper we

described a model of Hardware architecture that could be the backbone of the main hardware model of secured image transmission. The model contains two FPGA platforms namely the Xilinx Spartan 3E and Virtex 5 leading to a heterogeneous communication between them, and RS-232 cable for the medium of transmission. As a first step, from the host PC a pixel value of the digital image is read and it is being sent to the serial port of the PC in the MathWorks MATLAB software platform. Next form the serial port of PC the image data is transmitted to the DCE port of first FPGA board which performing as an encryption engine by executing stream cipher encrypting algorithm named RC4.The encrypted image data is then sent to the second FPGA platform which acts as the decryption engine by the means of execution of decryption part of RC4. The decrypted image data and the original image is viewed on a TFT monitor acting as a display unit. To accomplish this purpose, we had to customize a TFT interface controller from the board to the display unit which is another merit of our paper. For the sake of verification, we set up the TFT interface at the first board for observing the encrypted image.

Image Denoising

Residual Learning of Deep CNN For Image Denoising

Generally, training a deep CNN model for a specific task generally involves two steps: (*i*) network architecture design and (*i j*) model learning from training data. For network architecture design, we modify the VGG network to make it suitable for image denoising and set the depth of the network based on the effective patch sizes used in state-of-the-art denoising methods. For model learning, we adopt the residual learning formulation, and incorporate it with batch normalization for fast training and improved denoising performance. Finally, we discuss the connection between DnCNN and TNRD and extend DnCNN for several general image denoising tasks.

Network Depth

Set the size of convolutional filters to be 3 × 3 but remove all pooling layers. Therefore, the receptive field of Dn CNN with depth of d should be $(2d+1)\times(2d+1)$. Increasing receptive field size can make use of the context information in larger image region. For better tradeoff between performance and efficiency, one important issue in architecture design is to set a proper depth for DnCNN. It has been pointed out that the receptive field size of denoising neural networks correlates with the effective patch size of denoising methods. Moreover, high noise level usually requires larger effective patch size to capture more context information for restoration. Thus, by fixing the noise level $\sigma = 25$, we analyze the effective patch size of several leading denoising methods to guide the depth design of our DnCNN. In BM3D, the non-local similar patches are adaptively searched in a local widow of size 25 × 25 for two times, and thus the final effective patch size is 49×49. Like BM3D, WNNM uses a larger searching window and

performs non-local searching iteratively, resulting in a quite large effective patch size (361 × 361). MLP first uses a patch of size 39×39 to generate the predicted patch, and then adopts a filter of size 9×9 to average the output patches, thus its effective patch size is 47×47. The CSF and TNRD with five stages involves a total of ten convolutional layers with filter size of 7×7, and their effective patch size is 61×61. Table I summarizes the effective patch sizes adopted in different methods with noise level σ = 25. The effective patch size used in EPLL is the smallest, i.e., 36×36. It is interesting to verify whether DnCNN with the receptive field size like EPLL can compete against the leading denoising methods. Thus, for Gaussian denoising with a certain noise level, we set the receptive field size of DnCNN to 35 × 35 with the corresponding depth of 17. For other general image denoising tasks, we adopt a larger receptive field and set the depth to be 20.

Reducing Boundary Artifacts

In many low-level vision applications, it usually requires that the output image size should keep the same as the input one. This may lead to the boundary artifacts. In MLP, boundary of the noisy input image is symmetrically padded in the preprocessing stage, whereas the same padding strategy is carried out before every stage in CSF and TNRD. Different from the above methods, we directly pad zeros before convolution to make sure that each feature map of the middle layers has the same size as the input image. We find that the simple zero padding strategy does not result in any boundary artifacts. This good property is probably attributed to the powerful ability of the DnCNN.

Integration of Residual Learning and Batch Normalization for Image Denoising

The network can be used to train either the original mapping $F(\mathbf{y})$ to predict $\mathbf{x}$ or the residual mapping $R(\mathbf{y})$ to predict $\mathbf{v}$. According to, when the original mapping is more like an identity mapping, the residual mapping will be much easier to be optimized. Note that the noisy observation $\mathbf{y}$ is much more like the latent clean image $\mathbf{x}$ than the residual image $\mathbf{v}$ (especially when the noise level is low). Thus, $F(\mathbf{y})$ would be closer to an identity mapping than

$R(\mathbf{y})$, and the residual learning formulation is more suitable for image denoising. The average PSNR values obtained using these two learning formulations with/without batch normalization under the same setting on gradient-based optimization algorithms and network architecture. Note that two gradient-based optimization algorithms are adopted: one is the stochastic gradient descent algorithm with momentum (i.e., SGD) and the other one is the Adam algorithm. Firstly, we can observe that the residual learning formulation can result in faster and more stable convergence than the original mapping learning. In the meanwhile, without batch normalization, simple residual learning with conventional SGD cannot compete with the state-of-the-art denoising methods such as TNRD (28.92dB). We consider that the insufficient

performance should be attributed to the internal covariate shift caused by the changes in network parameters during training. Accordingly, batch normalization is adopted to address it. Secondly, we observe that, with batch normalization, learning residual mapping (the red line) converges faster and exhibits better denoising performance than learning original mapping (the blue line). Both the SGD and Adam optimization algorithms can enable the network with residual learning and batch normalization to have the best results. In other words, it is the integration of residual learning formulation and batch normalization rather than the optimization algorithms (SGD or Adam) that leads to the best denoising performance. One can notice that in Gaussian denoising the residual image and batch normalization are both associated with the Gaussian distribution. It is very likely that residual learning and batch normalization can benefit from each other for Gaussian denoising.1 This point can be further validated by the following analyses. • On the one hand, residual learning benefits from batch normalization. This is straightforward because batch normalization offers some merits for CNNs, such as alleviating internal covariate shift problem. One can see that even though residual learning without batch normalization has a fast convergence, it is inferior to residual learning with batch normalization.

On the other hand, batch normalization benefits from residual learning. Without residual learning, batch normalization even has certain adverse effect to the convergence. With residual learning, batch normalization can be utilized to speed up the training as well as boost the performance. Note that each mini-bath is a small set of images. Without residual learning, the input intensity and the convolutional feature are correlated with their neighbored ones, and the distribution of the layer inputs also rely on the content of the images in each training mini-batch. With residual learning, DnCNN implicitly removes the latent clean image with the operations in the hidden layers. This makes that the inputs of each layer are Gaussian-like distributed, less correlated, and less related with image content. Thus, residual learning can also help batch normalization in reducing internal covariate shift. To sum up, the integration of residual learning and batch normalization can not only speed up and stabilize the training process but also boost the denoising performance.

Deep Convolutional Neural Network for Inverse Problems in Imaging

The success of iterative methods consisting of filtering plus pointwise nonlinearities on normal-convolutional inverse problems suggests that CNNs may be a good fit for these problems as well. Based on this insight, we propose a new approach to these problems, which we call the *FBPConvNet* The basic structure of the FBPConvNet algorithm is to apply the discretized FBP from Section II-B (e.g., implemented by Matlab's iradon command) to the measurements and then use this as the input of a CNN which is trained to regress the FBP result to a suitable ground truth image. This approach applies in principle to all normal-convolutional

inverse problems, but we have focused in this work on CT reconstruction. We now describe the method in detail.

Filtered Back Projection

While it would be possible to train a CNN to regress directly from the measurement domain to the reconstruction domain, performing the FBP first greatly simplifies the learning. The FBP encapsulates our knowledge about the physics of the inverse problem and also provides a warm start to the CNN. For example, in the case of CT reconstruction, if the sonogram is used as input, the CNN must encode a change between polar and Cartesian coordinates, which is completely avoided when the FBP is used as input. We stress again that, while the FBP is specific to CT, Section II-B shows that efficient, direct inversions are always available for normal-convolutional inverse problems.

Deep Convolutional Neural Network Design

While we were inspired by the general form of the proximal update, (2), to apply a CNN to inverse problems of this form, our goal here is not to imitate iterative methods (e.g. by building a network that corresponds to an unrolled version of some iterative method), but rather to explore a state-of-the-art CNN architecture. To this end, we base our CNN on the U-net, which was originally designed for segmentation. For a diagram of our modified U-net, see Figure 2. For pseudocode of its operation, see the Appendix. There are several properties of this architecture that recommend it for our purposes.

Multilevel Decomposition: The U-net employs dyadic scale decomposition based on max pooling, so that the effective filter size in the middle layers is larger than that of the early and late layers. This is critical for our application because the filters corresponding to $H*H$ (and its inverse) may have non-compact support, e.g. in CT. Thus, a CNN with a small, fixed filter size may not be able to effectively invert $H*H$. This decomposition also has a nice analog to the use of multiresolution wavelets in iterative approaches.

Multichannel Filtering: U-net employs multichannel filters, such that there are multiple feature maps at each layer. This is the standard approach in CNNs to increase the expressive power of the network. The multiple channels also have an analog in iterative methods: In the ISTA formulation, we can think of the wavelet coefficient vector $\mathbf{a}$ as being partitioned into different channels, with each channel corresponding to one wavelet subband. Or, in ADMM, the split variables can be viewed as channels. The CNN architecture greatly generalizes this by allowing filters to make arbitrary combinations of channels.

Residual Learning: As a refinement of the original U-net, we add a skip connection between input and output, which means that the network actually learns the difference between input and output. The vanishing gradient problem during training. Thisyields a noticeable increase in performance compared to the same network without the skip connection.

Implementation Details:

We made two additional modifications to U-net. First, we use zero-padding so that the image size does not decrease after each convolution. Second, we replaced the last layer with a convolutional layer which reduces the 64 channels to a single output image. This is necessary because the original U-net architecture results in two channels: foreground and background.

Computational Complexity

The computational cost of the FBP is dominated by the back projection, rather than the filtering. For an $N \times N$ image and a $M \times V$ sinogram, the cost of the back projection scales with $O(N2\ MV)$ in the worst case, though this can be reduced to $O(N2\ V)$ by considering a fixed-size discretization kernel (this is the case with the implementation we use). The basic operations in the CNN are convolutions, additions, application of the ReLU function, upsampling, downsampling, and local maximum filtering. The operation count is dominated by the convolutions, which are performed in the space domain because the kernel is small (3×3 in our case). More specifically, for an $N \times N$ input, $K \times K$ filters, R filters per layer, and L layers, the cost of evaluating the CNN grows like $O(N2\ K2\ R2\ L)$. The storage for the network is only dependent on the size of the filters and biases. Therefore, this can be summarized as $O(LK2R2)$. During training, the computation is dominated by the chain rule calculations in the error-backpropagation algorithm. These are essentially the same procedures as the forward (evaluation) path except in reverse. Thus, the computational complexity of training linearly scales with the number of network variables and training dataset size. The storage for the training phase is larger than that of the evaluation because it demands saving feature maps for each layer of the network during error back propagation. Usually, the memory complexity becomes $O(N2\ RL)$ in order to include all the feature maps in the process. Both the evaluation and training of the CNN is performed on the GPU. Each operation in the CNN is simple and local, ideal for GPU-based paralellization. For example, convolutions in the network are driven by GPU calculation supported by the MatConvNet toolbox or cuDNN (NVIDIA).

Color Balance and Fusion for Underwater Image Enhancement

Multi-Scale Fusion:

In this work, the multi-scale fusion principles are built to propose a single image underwater dehazing solution. Image fusion has shown utility in several applications such as

image compositing, multispectral video enhancement, defogging and HDR imaging. Here, we aim for a simple and fast approach that is able to increase the scene visibility in a wide range of underwater videos and images. The framework builds on a set of inputs and weight maps derived from a single original image. However, those ones are specifically chosen in order to take the best out of the white-balancing method introduced in the previous section. In particular, a pair of inputs is introduced to respectively enhance the color contrast and the edge sharpness of the white-balanced image, and the weight maps are defined to preserve the qualities and reject the defaults of those inputs, i.e. to overcome the artifacts induced by the light propagation limitation in underwater medium. This multi-scale fusion significantly differs from our previous fusion-based underwater dehazing approach published at IEEE CVPR . To derive the inputs from the original image, our initial CVPR algorithm did assume that the backscattering component (due to the artificial light that hits the water particles and is then reflected back to the camera) has a reduced influence. This assumption is generally valid for underwater scenes decently illuminated by natural light, but fails in more challenging illumination scenarios, as revealed in the results section. In contrast, this paper does not rely on the optical model and proposes an alternative definition of inputs and weights to deal with severely degraded scenes. our underwater dehazing technique consists in three main steps: inputs derivation from the white balanced underwater image, weight maps definition, and multi-scale fusion of the inputs and weight maps.

Inputs of the Fusion Process

Since the color correction is critical in underwater, we first apply our white balancing technique to the original image. This step aims at enhancing the image appearance by discarding unwanted color casts caused by various illuminants. In water deeper than 30 ft, white balancing suffers from noticeable effects since the absorbed colors are difficult to be recovered. As a result, to obtain our first input we perform a gamma correction of the white balanced image version. Gamma correction aims at correcting the global contrast and is relevant since; in general, white balanced underwater images tend to appear too bright. This correction increases the difference between darker/lighter regions at the cost of a loss of details in the under-/over-exposed regions. To compensate for this loss, we derive a second input that corresponds to a sharpened version of the white balanced image. Therefore, we follow the unsharp masking principle, in the sense that we blend a blurred or unsharp (here Gaussian filtered) version of the image with the image to sharpen. The typical formula for unsharp masking defines the sharpened image S as $S = I + \beta(I - G * I)$, where I is the image to sharpen (in our case the white balanced image), $G * I$ denotes the Gaussian filtered version of I, and β is a parameter. In practice, the selection of β is not trivial. A small β fails to sharpen I , but a too large β results in over-saturated regions, with brighter highlights and darker shadows. To

circumvent this problem, we define the sharpened image S as follows: $S = (I + N\{I - G * I\})/2$, with $N\{.\}$ denoting the linear normalization operator, also named histogram stretching in the literature. This operator shifts and scales all the color pixel intensities of an image with a unique shifting and scaling factor defined so that the set of transformed pixel values cover the entire available dynamic range. The sharpening method defined is referred to as *normalized unsharp masking* process in the following. It has the advantage to not require any parameter tuning, and appears to be effective in terms of sharpening. This second input primarily helps in reducing the degradation caused by scattering. Since the difference between white balanced image and its Gaussian filtered version is a highpass signal that approximates the opposite of Laplacian, this operation has the inconvenient to magnify the high frequency noise, thereby generating undesired artifacts in the second input. The multi-scale fusion strategy described in the next section will be in charge of minimizing the transfer of those artifacts to the final blended image.

Weights of the Fusion Process

The weight maps are used during blending in such a way that pixels with a high weight value are more represented in the final image. They are thus defined based on a number of local image quality or saliency metrics.

Laplacian contrast weight (WL) estimates the global contrast by computing the absolute value of a Laplacian filter applied on each input luminance channel. This straightforward indicator was used in different applications such as tone mapping and extending depth of field since it assigns high values to edges and texture. For the underwater dehazing task, however, this weight is not sufficient to recover the contrast, mainly because it can not distinguish much between a ramp and flat regions. To handle this problem, we introduce an additional and complemntary contrast assessment metric. **Saliency weight** (WS) aims at emphasizing the salient objects that lose their prominence in the underwater scene. This computationally efficient algorithm has been inspired by the biological concept of center-surround contrast. However, the saliency map tends to favor highlighted areas (regions with high luminance values). To overcome this limitation, we introduce an additional weight map based on the observation that saturation decreases in the highlighted regions.

Saturation weight ($WSat$) enables the fusion algorithm to adapt to chromatic information by advantaging highly saturated regions. This weight map is simply computed (for each input Ik) as the deviation (for every pixel location) between the Rk, Gk and Bk color channels and the luminance. In practice, for each input, the three weight maps are merged in a single weight map as follows. For each input k, an aggregated weight map Wk is first obtained by summing up the three WL, WS, and $WSat$ weight maps. The K aggregated maps are then normalized on a pixel-per-pixel basis, by dividing the weight of each pixel in each map by the sum of the weights

of the same pixel over all maps. Formally, the normalized weight maps. Note that, in comparison with our previous work, we limit ourselves to these three weight maps only, and we do not compute the exposedness weight map anymore. In addition to reducing the overall complexity of the fusion process, we have observed that, when using the two inputs proposed in this paper, the exposedness weight map tends to amplify some artifacts, such as ramp edges of our second input, and to reduce the benefit derived from the gamma corrected image in terms of image contrast. We explain this observation as follows. Originally, in an exposure fusion context, the exposedness weight map had been introduced to reduce the weight of pixels that are under- or over-exposed. Hence, this weight map assigns large (small) weight to input pixels that are close to (far from) the middle of the image dynamic range. In our case, since the gamma corrected input tends to exploit the whole dynamic range, the use of the exposedness weight map tends to penalize it in favor of the sharpened image, thereby inducing some sharpening artifacts and missing some contrast enhancements.

Semantic Segmentation and Classification of Cell Membranes and Nuclei in Breast Cancer Evaluation

In this study, the dataset mainly consisted of 158 WSIs, which were acquired using a Hamamatsu NanoZoomer C9600 scanner. Out of 158 WSIs, 79 were stained using haematoxylin & eosin (H&E), and the other 79 were stained using HER2 monoclonal antibody. The size of each WSI was 100,000×80,000 pixels (width×height). The WSIs could be viewed at 4× to 40× magnification. In our study, we used only the 79 HER2 monoclonal antibody-stained WSIs to divide the dataset into two subsections, i.e., training (51 WSIs) and testing (28 WSIs). A total of 752 images (188 for each score, i.e., 0, 1+, 2+, 3+) at 40× magnification were cropped (using MATLAB *imcrop*() function) from the 79 WSIs. The size of the each cropped image was kept as 2048×2048 pixels (width×height). Noise during image grabbing was automatically removed by the Hamamatsu NanoZoomer image-grabbing software. The rest of the biological and acquisition-related noise was removed using morphological operations and an adaptive filter, respectively.

Patch Selection

The HER2 image patches of size 251 × 251 were cropped from images of size 2048×2048 pixels based on the sequential window movement technique. In this technique, the window will move from left to right and up and down over the images. We considered zero or very negligible overlapping regions in our image patches. Around the boundary areas, where the image size did not match the patch size, patches were not included in our experiment.

Convolutional Layer

The convolution layer was used to convolve the image patches with the kernels. Here, the resultant image's pixel value is the sum of products of the image pixels of local receptive fields and filter coefficients. The convolution operation is done across $(\omega - m + 1) \times (\omega - m + 1)$ pixels. Now the resultant image *Convout* size is $(\omega - m + 1) \times (\omega - m + 1)$.

Pooling Layers

The principle of pooling layers reduces the feature map dimensionality for computational efficiency. Hence, pooling layers are often called down-sampling layers. The maxpooling and spatial pyramid pooling layers are the most used down-sampling layers. Down-sampling layers make features transition-independent. Three types of pooling operations were used in our proposed framework, i.e., max-pooling, spatial pyramid pooling and up-sampling/up-pooling. The max-pooling layer was used to perform spatial invariance using feature map resolution reduction. The max operation of max-pooling layer helped in aggregating the features from the spatial regions. The spatial pyramid sampling/pooling was used to divide the image patches into areas to discern local information of the image. In this pooling operation, the partition was considered in itself, and its features were combined in some way, first for each partition and then for the combination of those partitions as a global representation of the image The properties of up-sampling layers are completely opposite to down-sampling layers. It is a reverse operation of max-pooling layers. Up-sampling of layers was used to enlarge the input image and densify sparse activations. In the deconvolution layer, the up-sampling helped in reconstructing the original image and preserved the structure of a stimulus.

Deconvolution Layer

A deconvolution layer was used after an up-sampling layer. This layer densified the sparse activations through multiple learned filters and convolutionlike operations. Unlike a convolution layer, the deconvolution layer connected single-input activation with multiple outputs. The deconvolution layer gave a dense and enlarged action map of an input image. The learned filters in this layer were mainly responsible for reconstructing the shape of an input image. A hierarchical structure of deconvolution layers was formed to save the shape information of the input image. The lower-layer filters were responsible for capturing shape information. Through this procedure, deconvolution layers assisted in semantic segmentation.

Fully Connected (FC) Layer

The FC layer was used as a final layer or classification layer. In our *Her2Net* framework, the FC layer's height and width of each blob was assigned to 1.

Dropout Layer

Dropout is a regularization technique which increases the accuracy of a model by reducing overfitting and preventing complex co-adaption on training data. The dropout neurons do not contribute in the forward pass or back-propagation.

Softmax Layer

The softmax layer was used to reduce cross-entropy in a multi-class problem.

Long Short-Term Memory (LSTM)

LSTM, a recurrent network, computes a new hidden state where new input and previously hidden states are given. Unlike other recurrent neural networks (RNNs), LSTM does not suffer from vanishing gradients or the phenomena of exploding. LSTM is widely used in speech recognition and handwriting recognition. The goal of using LSTM in *Her2Net* was to employ pixel details from the multiple frames in making the semantic segmentation and classification cases. The memory cells of LSTM were used for maintaining long-range relationships with the rest of the network. Our single-cell LSTM connection. The one-cell LSTM consisted of input, memory cell, output and the three gates (i.e., input, forget and output gates). These gates used logistic functions for computing the activation of the LSTM. In our method, these gates were considered as conventional artificial neurons. Each gate possessed its own weight and bias values, which were basically the output of the previous layers outside LSTM. The input gate controlled a mechanism on which a value flowed into a memory. The forget gate was used to keep a value in memory. Finally, the output gate was used to control the activation of LSTM based on the total parameters.

Single-Image Super-Resolution Based on Adaptive Edge-Preserving Smoothing Regularization

Learning the Self-Example Sub-Dictionaries

Learning a representative dictionary is a critical issue in sparse representation modeling. In the proposed REPS-SR model, a series of sub-dictionaries are learned to code various image structures. In contrast to the ASDS method, we learn the sub-dictionaries from the image itself which makes the proposed REPS-SR method more adaptive for different images. In the proposed method, we first extract image patches from image $\boldsymbol{x}$. In order to learn a series of sub-dictionaries to code various local image structures, a set of image patches will be selected to be involved in the dictionary learning process. The image patch is selected when its intensity variance is greater than a threshold. This patch selection criterion tries to exclude the smooth patches from training and guarantees that only these patches including a certain amount of edge structures are used to learn the dictionary. Suppose that M image patches $X = [\boldsymbol{x}1\ ,\ \boldsymbol{x}2, \ldots \boldsymbol{x}M]$ are selected, we have to learn K sub-dictionaries from X so that the most

relevant sub-dictionaries can be selected adaptively for the local image patches. The image patches are clustered into *K* clusters and we learn a compact dictionary from each cluster. To make the dictionaries representative for image edges and local structures, the high-pass filtering is used to process the entire set of image patches before clustering. The *K*-means algorithm is adopted to cluster the high-pass filtered image patches into *K* clusters. Since learning an over-complete dictionary is computationally costly; a compact dictionary is learned using PCA because the patches in a cluster tend to have similar structures and redundancy information.

$\mathbf{S}_k$ denotes the sub-dataset after high-pass filtering and *K*-means algorithm, and $\_k$ denotes the co-variance matrix of $\mathbf{S}_k$.

REPS-SR Model

The REPS-SR method can be used to solve the single-image SR problem. Since image patches often contain repetitive patterns, such nonlocal redundancy is very helpful for improving the reconstruction performance of images. The nonlocal self-similarity is used as a regularization term to provide a good estimation of $\boldsymbol{\alpha}i$. Meanwhile, considering that image edges convey abundant image information, we use the image edges as a regularization term to constrain the image SR model. Here, an EPS regularization is designed to be incorporated into the sparse model. In the REPS-SR model, the reconstructed image $\_\boldsymbol{x}$ at each iteration is regarded as the guidance image and the input filtering image. During iterative processing, the proposed EPS regularization term constrains the reconstructed HR image close to the filtered

image which is an edge-preserving smoothing image. The reconstructed image edges can be preserved, and the noise can be reduced. Furthermore, the edges of reconstructed images can be enhanced after multiple iterations.

Adaptive Smoothing-Aware Factor

In REPS-SR model, the regularization parameter $\lambda 1$ controls the tradeoff between the fidelity to data and the nonlocal regularization, and $\lambda 2$ controls the smoothness of the solution. A suitable choice of regularization parameter is beneficial to obtain a global optimal solution and improve the convergence rate. Dong *et al*. proposed to use the MAP estimator to adaptively determine the regularization parameter $\lambda 1$ which gets good SR performance. In the proposed method, the determination of $\lambda 1$ uses the same scheme as the NCSR model. To make our method more competitive, we propose to adaptively obtain the smoothing-aware factor $\lambda 2$. In the SR problem, a large value of the regularization parameter will be used when the number of LR images is small or the fidelity of the observed data is low, which might be caused by registration error or noise. On the other hand, a small parameter will lead to a good solution of

the SR problem when the noise level is small. It means that a higher level of noise level should be corresponding to a larger value of regularization parameter. In the proposed method, we adaptively determine the smoothing-aware factor $\lambda 2$ by regarding the factor $\lambda 2$ as a function of unknown noise level. In this paper, the noise level of LR image is estimated based on the selected weak textured patches using PCA. Many experiments were conducted to verify the generality of the theory that a high level of noise corresponds to a large value of $\lambda 2$. Different levels of Gaussian noise are added to various images, and different values of $\lambda 2$ that range from 0 to 2 are used in our experiments to obtain many SR results. Table I shows the average experimental results when noise level ranges from 0 to 10 and $\lambda 2$ ranges from 0 to 1.2 with six test images.

3D Face Reconstruction with Geometry Details from a Single Image

Low-Dimensional Models

Human faces have similar global characteristics, for example the location of main facial features such as eyes, nose and mouth. From a perception perspective, it has been shown that a face can be characterized using a limited number of parameters. The low dimensionality of the face space allows for effective parametric face representations that are derived from a collection of sample faces, reducing the reconstruction problem into searching within the parameter space. A well-known example of such representations is the 3DMM proposed in, which has been used for various face processing tasks such as reconstruction, recognition, face exchange in images and makeup suggestion. Low dimensional representations have also been used for dynamic face processing. To transfer facial performance between individuals in different videos, Vlasic et al. develop a multilinear face model representation that separately parameterizes different face attributes such as identity, expression, and viseme. In the computer graphics industry, facial animation is often achieved using linear models called blend shapes, where individual facial expressions are combined to create realistic facial movements. The simplicity and efficiency of blend shapes models enable real-time facial animation driven by facial performance captured from RGBD cameras and monocular videos. When using low dimensional face representations derived from example face shapes, the example dataset has strong influence on the resulting face models. For instance, it would be difficult to reconstruct a facial expression that deviates significantly from the sample facial expressions. In the past, during the development of face recognition algorithms, various face databases have been collected and made publicly available.

Shape-From-Shading

Shape-from-shading (SFS) is a computer vision technique that recovers 3D shapes from their shading variation in 2D images. Given the information about illumination, camera

projection, and surface reflectance, SFS methods can recover fine geometric details that may not be available using low-dimensional models. On the other hand, SFS is an ill-posed problem with potentially ambiguous solutions. Thus for face reconstruction, prior knowledge about facial geometry must be incorporated to achieve reliable results. For example, symmetry of human faces has been used by various authors to reduce the ambiguity of SFS results. Another approach is to solve the SFS problem within a human face space, using a low-dimensional face representation. Other approaches improve the robustness of SFS by introducing an extra data source, such as a separate reference face, as well as coarse reconstructions using multiview stereo or unconstrained photo collections. We adopt a similar approach which builds an initial estimation of the face shape and augment it with fine geometric details using SFS. Our initial face estimation combines coarse reconstruction in a low-dimensional face space with refinement of medium-scale geometric features, providing a more accurate initial shape for subsequent SFS processing.

Coarse Face Modeling

Preprocessing

The FACEWAREHOUSE dataset contains head meshes of 150 individuals, each with 47 expressions. All expressions are represented as meshes with the same connectivity, each consisting of 11510 vertices. The BFM2009 dataset contains 200 face meshes, and each mesh consists of 53490 vertices. To combine the two datasets, we first mask the face region on the head mesh from FACEWAREHOUSE to extract a face mesh and fill the holes in the regions of eyes and mouth, to obtain a simply connected face mesh consisting of 5334 vertices. Afterwards, we randomly sample the parameter space for BFM2009 to generate 150 neutral face models and deform the average face model from FACEWAREHOUSE to fit these models via nonrigid registration. Then we transfer the other 46 expressions of the FACEWAREHOUSE average face model to each of the 150 deformed face models based on the method. In this way, we construct a new dataset containing 300 individuals (150 from BFM2009 and 150 from FACEWAREHOUSE), each with 47 expressions. We perform Procrustes alignment for all the face meshes in the dataset. Moreover, BFM2009 provides 199 principal components to span the surface albedo space, but these principal albedo components cannot be used for our new dataset directly due to different mesh connectivity. Thus, we transfer their albedo information to the new mesh representation using the correspondence identified in the nonrigid registration, to construct 199 principal albedo components for our dataset. These principal components will be used.

Bilinear Face Model

We collect the vertex coordinates of all face meshes into a third-order data tensor and perform 2-mode SVD reduction along the identity mode and the expression mode, to derive a bilinear face model that approximates the original data set. In detail, the bilinear face model is represented as a mesh with the same connectivity as those from the data set to construct a coarse face, we align 3D landmarks on the bilinear face model with corresponding 2D landmarks from the input image. First, we preprocess the bilinear face mesh to manually label 68 landmark vertices. Given an input image, we detect the face as well as its corresponding 68 landmarks using the method

Landmark Vertex Update

The landmark vertices on the face mesh are labeled based on the frontal pose. For non-frontal face images, the detected 2D landmarks along the face silhouette may not correspond well with the landmark vertices. Thus, after each camera parameter optimization step, we update the silhouette landmark vertices according to the rotation matrix **R**, while keeping the internal landmark vertices (e.g., those around the eyes, the nose, and the mouth) unchanged. Like we preprocess the original face mesh to derive a dense set of horizontal lines that connect mesh vertices and cover the potential silhouette region from a rotated view. Given a rotation matrix **R**, we select from each horizontal line a vertex that lies on the silhouette, and project it onto the image plane according to the camera parameters **_**, **R**, **t**. These projected vertices provide an estimate of the silhouette for the projected face mesh. Then for each 2D silhouette landmark, its corresponding landmark vertex is updated to the silhouette vertex whose projection is closest to it.