Fast Algorithms for Inventory Based Speech Enhancement

Robert M. Nickel and Tomohiro Sugimoto
Department of Electrical Engineering
Bucknell University
Lewisburg, PA 17837
robert.nickel@bucknell.edu

Xiaoqiang Xiao
Department of Electrical Engineering
The Pennsylvania State University
University Park, PA 16802
xxx106@psu.edu

1. Introduction

A significant number of the algorithms that are currently employed in commercially available single-channel speech enhancement products are *waveform filtering* based methods. Waveform filtering implies that an appropriately chosen filter[1] is applied to the incoming noisy speech data in order to change or smooth the shape the resulting filtered waveform. The objective of the filtering is usually to either improve the *perceptual quality* of the output -or- to improve the *recognition rate* of a subsequently used speech recognition system. Prominent examples of waveform processing are the Wiener filtering extensions proposed by McAulay & Malpass (1980) and Ephraim & Malah (1984). Other examples include schemes that employ wavelets by Hu & Loizou (2004) and modifications of the iterative Wiener filter and the Kalman filter by Mouchtaris et al. (2007). Also related are the spectral subtraction method developed by Boll in 1979 (see the text by Deller et al. (1993)) and its powerful extension, the *multiband spectral subtraction* described in the text by Loizou (2007).
The success of many waveform filtering based methods is due to their relative (computational) simplicity and robustness. A disadvantage of filtering based methods, however, is that they are never able to completely remove the noise. They are usually aiming to achieve a reasonable tradeoff between a desired reduction of the noise and an undesired but inadvertent distortion of the targeted signal.
The search for a denoising paradigm that, at least in theory, allows for potentially "perfect" enhancement has motivated many researchers to study model based denoising methods. In model based denoising a parametric model for a speech signal (which may be deterministic or stochastic in nature) is used instead of a general waveform model. A popular choice

[1] The employed filters are typically linear but potentially time-variant.

for a speech model in this context is the *harmonic plus noise model* (HNM) which was studied amongst others by Zavarehei et al. (2007). Related is also the work by Zhao & Kleijn (2007) on the modelling and estimation of speech and noise gains via hidden Markov models. Codebooks of linear predictive coefficients and their employment for speech denoising within a maximum-likelihood framework were studied by Srinivasan et al. (2006). A minimum mean square error approach for denoising that relies on a combined stochastic and deterministic speech model was studied by Hendriks et al. (2007).

The model based speech denoising method discussed in this chapter was proposed by Xiao et al. (Aug. 2008) and (Apr. 2009). It is inspired by the increasing success of *inventory based speech synthesis systems* as discussed in a review paper by O'Shaughnessy (2007). In this work it is assumed that speaker enrollment and noise enrollment are feasible. The speaker enrollment procedure provides training data that can be appropriately clustered and used as an inventory for a "clean" speech signal model. The inventory based denoising is supported by a statistical analysis of the speech signal under clean and noisy conditions.

One of the disadvantages of this method in comparison to other model based approaches is its high computational complexity. The procedure requires, in its originally proposed form, a very large number of floating point multiplications. The bottleneck of the procedure can be found in correlation operations that need to be carried out over large data records. In this chapter we are describing a modification of the original approach that incorporates a *fast algorithm* for the denoising stage. The proposed method substantially reduces the amount of necessary multiplications via the employment of a set of number theoretic transforms (NTTs, Blahut (1987)). Number theoretic transforms allow the computation of correlations in fixed-point arithmetic with a substantial reduction in multiplications. However, NTTs come generally at the expense of reduced computational accuracy due to the required signal quantization. We present an approach that balances a dramatic reduction in computational complexity with only a very slight reduction in perceptual denoising performance.

The chapter is divided into two main sections. Section 2 provides a summary of the proposed denoising paradigm. A full and detailed description of the method is beyond the scope of this chapter. The interested reader will find it in the two papers by Xiao et al. (Aug. 2008) and (Apr. 2009). Section 3 focuses explicitly on the parts of the procedure that can be improved with a fast algorithm.

2. The Denoising Paradigm

The method proposed by Xiao et al. (Apr. 2009) can be divided into three main tasks: (i) a system training task, (ii) the signal preprocessing task, and (iii) the signal denoising task. The main contribution of this work can be found in the modification of the signal denoising task. For the reader's convenience, however, we are providing a cursory overview of the entire system in this section. The description follows closely in structure and notation with that of the original paper by Xiao et al. (Apr. 2009). Many key details, however, are omitted here. The interested reader may want to consult the original paper for a comprehensive presentation.

The system training task consists of the development of a *speech waveform inventory*, two *mel-frequency cepstral coefficient* (MFCC) codebooks (under clean and noisy conditions), and a *hidden Markov model* (HMM). The HMM is used to model the codeword transition statistics under clean and noisy conditions. The system training task is discussed in some greater detail in section 2.1.

The procedures of the signal preprocessing task are adjusted according to the expected noise type. Three different noise types are considered in the original paper. They are white

noise, colored noise, and non-stationary noise. No preprocessing is performed in the case of white noise. Stationary colored noise requires preprocessing with a *prewhitening filter.* Non-stationary noise is preprocessed with a combination of a short-time power spectral estimator (via *harmonic tunnelling*, Ealey et al. (2001)) and subsequent *Wiener filtering.*
Lastly, the speech denoising task combines the results of the preprocessing with the results of a *state sequence computation* from the trained HMM as described in section 2.1. Suitable sections from the speech inventory are chosen through an *inventory unit selection scheme* and are then concatenated to form the targeted denoised speech signal (see section 2.2). The inventory unit selection scheme constitutes the computational bottleneck of the procedure. Most other components of the method have a computational complexity that is comparable to that of other model based methods. The complexity of the inventory unit selection scheme, however, dominates the overall processing requirement by an order of magnitude. The fast processing algorithm presented in this work focuses therefore exclusively on the inventory unit selection. It is comprehensively described in section 3.
Throughout this chapter we are using a mathematical notation that is consistent with the one introduced in the paper by Xiao et al. (Apr. 2009). At the *denoising* stage we assume that we observe a signal $x[n]$ which consists of speech $s[n]$ that is uttered by the enrolled speaker and is distorted by zero mean additive noise $v[n]$, i.e. $x[n] = s[n] + v[n]$. At the *training* stage we use $\hat{s}[n]$ to, similarly, denote the *speaker enrollment data.* System training is done off-line from speaker-specific pre-recorded *clean* training signals. For simplicity we assume that all training records of speech are concatenated into one long training sequence $\hat{s}[n]$.
An accurate description of the enhancement procedure requires the definition of speech *units* or *frames.* We represent a unit as a vector of N successive samples of a signal:

$$\mathbf{s}_n = [\ s[n-L]\ \ s[n-L+1]\ \ \ldots\ \ s[n-L+N-1]\ ]^{\mathrm{T}}. \tag{1}$$

Note that in section 3 we employ signal segments of a different length, i.e. segments with a processing block-length of K (with $K > N$, see equations (10) and (11)). The amount of overlap between adjacent frames is controlled by a step size L. If i denotes a unit (or frame) index then the associated vector is written as $\mathbf{s}_{iL}$. Symbols $\mathbf{x}_n$, $\mathbf{v}_n$, and $\hat{\mathbf{s}}_n$ are defined analogously to equation (1). Symbol $\mathbb{S}$ is used to denote our *speech-waveform-unit inventory.* Set $\mathbb{S}$ consists of all clean training data frames $\hat{\mathbf{s}}_n$ ($\forall n$, i.e. with a step size of one) with the exception of data frames that are entirely silent. Data frames are considered entirely silent if the total frame energy falls below a certain minimal level.
The fundamental paradigm behind the considered denoising method is quite simple: find a mapping $\mathbf{x}_{iL} \to \hat{\mathbf{s}}_{n(i)}$ that associates a specific inventory frame $\hat{\mathbf{s}}_{n(i)}$ to every observed noisy frame $\mathbf{x}_{iL}$. The complexity of the method arises from the fact that this mapping is generally not fixed, but time-variant and context dependent. A resulting denoised signal $\tilde{s}[n]$ is obtained by "concatenating" the found frames $\hat{\mathbf{s}}_{n(i)}$ via a *sinusoidal model* based resynthesis technique. The employed resynthesis technique is similar to the one described in the text by Quatieri (2002). Please refer to the original paper by Xiao et al. (Apr. 2009) for the details.

2.1. System Training and State Sequence Estimations

The system training stage is used to achieve two separate goals: (1) to provide the denoising procedure with an *inventory* of available speech units and (2) to generate a *hidden Markov model* that describes transition statistics within the inventory. An illustration of the inventory design procedure is shown in figure 1. All inventory elements $\hat{\mathbf{s}}_n$ that belong to a similar *phonemic*

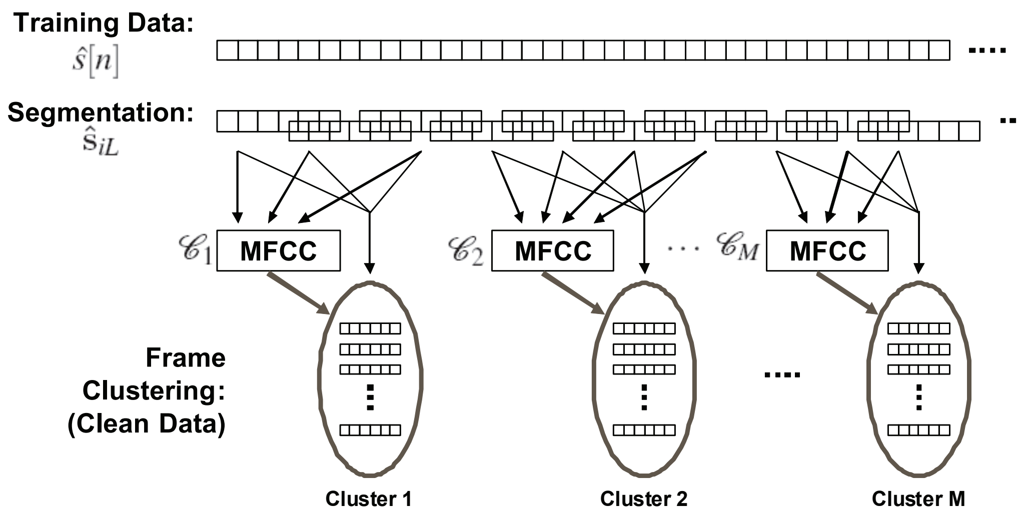


Fig. 1. An illustration of the data clustering method used by the denoising procedure by Xiao et al. (Apr. 2009). Training data is segmented, MFCC coefficients are extracted, and frames with "similar" coefficient vectors are lumped into one of $M = 50$ inventory clusters.

function[2] are grouped into the same cluster. The purpose of the grouping is to be able to study the statistical properties of the group as a whole and then apply a resulting statistical description in the denoising process. The procedure requires the construction of two *mel-frequency cepstral coefficient* (MFCC) codebooks: a clean MFCC codebook $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_M\}$ and a noisy MFCC codebook $\widehat{\mathcal{C}} = \{\widehat{\mathcal{C}}_1, \widehat{\mathcal{C}}_2, \ldots, \widehat{\mathcal{C}}_M\}$. The codebooks contain the average of the MFCC vectors of all clean/noisy inventory units within a respective cluster. The details of the clustering procedure and the inventory design are omitted here. Please consult the paper by Xiao et al. (Aug. 2008) for a comprehensive description. An illustration of the noisy codebook design procedure is shown in figure 2. To maintain compatibility with the notation introduced in Xiao et al. (Aug. 2008) we will refer to the resulting cluster sets of inventory vectors $\hat{\mathbf{s}}_n$ with $\mathbb{K}_k$ for $k = 1, 2, \ldots M$.
Given the clean and noisy MFCC codebook vectors for each inventory cluster it becomes possible to estimate the cluster transition statistics for the given speaker. An illustration of the considered statistical description after Xiao et al. (Aug. 2008) is shown in figure 3. We use $\hat{\mathbf{s}}_n \to k$ to indicate the cluster membership of inventory frame $\hat{\mathbf{s}}$ with cluster k. Similarly, we define $\hat{\mathbf{x}}_{iL} \to j$ to indicate that the incoming noisy frame $\hat{\mathbf{x}}_{iL}$ is vector quantized via the noisy codebook $\widehat{\mathcal{C}}$ into cluster j. With a simple counting process we can estimate the first-order temporal *state transition probabilities,* i.e.

$$P_{k,j} = \mathrm{Prob}[\, \hat{\mathbf{s}}_{(i+1)L} \to j \,|\, \hat{\mathbf{s}}_{iL} \to k \,]. \tag{2}$$

Similarly, we can convert our sequence of noisy training frames $\hat{\mathbf{x}}_{iL}$ into an *observation code sequence.* Again, with a counting process we can estimate the noise induced *observation probabilities* jointly from our clean and noisy training data:

$$Q_{k,j} = \mathrm{Prob}[\, \hat{\mathbf{x}}_{iL} \to j \,|\, \hat{\mathbf{s}}_{iL} \to k \,]. \tag{3}$$

[2] We are using the term *phonemic function* in reference to a general, function carrying unit of a language. The group *may* or *may not* match with an actual *phoneme* defined for that language.

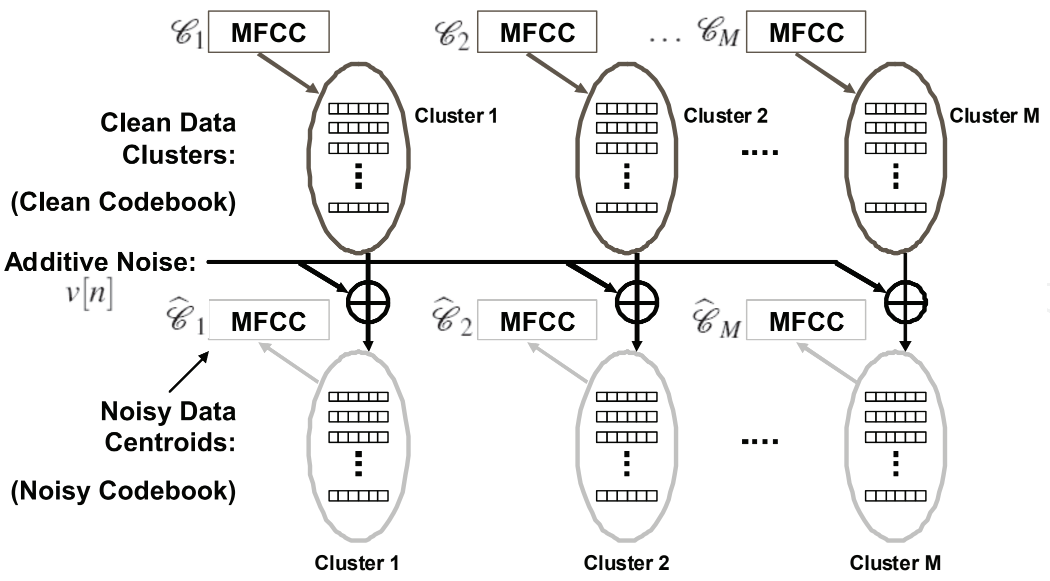


Fig. 2. An illustration of the generation of the noisy MFCC codebook as proposed by Xiao et al. in (Aug. 2008) and (Apr. 2009). Training noise is added to the elements of the clean inventory. The noisy MFCC codebook arises from the average of the MFCC vectors computed from the respective distorted signals within each clean cluster.

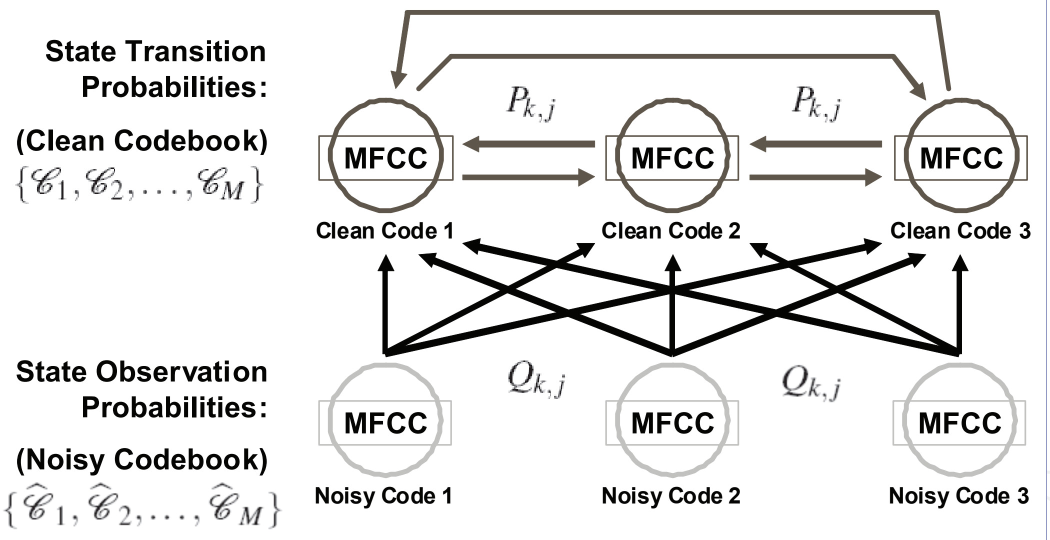

State Transition Probabilities:
(Clean Codebook)
$\{\mathscr{C}_1, \mathscr{C}_2, \ldots, \mathscr{C}_M\}$
$P_{k,j}$
$P_{k,j}$
MFCC
Clean Code 1
Clean Code 2
Clean Code 3
State Observation Probabilities:
(Noisy Codebook)
$\{\widehat{\mathscr{C}}_1, \widehat{\mathscr{C}}_2, \ldots, \widehat{\mathscr{C}}_M\}$
$Q_{k,j}$
$Q_{k,j}$
Noisy Code 1
Noisy Code 2
Noisy Code 3

Fig. 3. An illustration of the statistical description of the considered cluster membership after Xiao et al. (Aug. 2008). The observation of a "noisy" code is statistically related to the "true" cluster membership of the underlying clean signal segment.

The transition probabilities $P_{k,j}$ and $Q_{k,j}$ are both used in the denoising process. The statistical description enables us to define an "optimal" sequence $k_{opt}(i)$ of cluster memberships for incoming testing frames $\mathbf{x}_{iL}$. The sequence is optimal in the sense that the "most likely" inventory element $\hat{\mathbf{s}}_{n(i)}$ to represent the denoised frame for $\mathbf{x}_{iL}$ is found in set $\mathbb{K}_{k_{opt}(i)}$. Again, the details of how to find the sequences $k_{opt}(i)$ are omitted. A comprehensive description is found in the paper by Xiao et al. (Aug. 2008).

2.2. Speech Denoising

After completion of the described system training we can begin to denoise new incoming signals $x[n]$. The denoising procedure can be broken down into 4 separate steps: (1) the estimation of the "optimal" cluster membership sequence $k_{opt}(i)$ (as discussed in the previous section), (2) the preprocessing, i.e. prewhitening, of the noisy signal $x[n]$, (3) the identification of the best match for each $\mathbf{x}_{iL}$ in $\mathbb{K}_{k_{opt}(i)}$, i.e. the *intra cluster frame matching,* and (4) the "concatenation" of the resulting inventory frames to resynthesize the targeted denoised signal.
As indicated in the previous section, we will omit the details of the $k_{opt}(i)$-sequence estimation. Sequence $k_{opt}(i)$ can be computed fast and efficiently via the *Viterbi algorithm,* as described in the paper by Xiao et al. (Aug. 2008). We also omit most of the details of the data preprocessing, as they have been comprehensively described in the same paper. We will, however, briefly describe the underlying principles of the *intra cluster frame matching* since it is the target of the proposed fast algorithm described in section 3.
In a first step we define a similarity measure between a noisy frame $\mathbf{x}_{iL}$ and an inventory element $\hat{\mathbf{s}}_n$. The choice of the similarity measure proposed in the paper by Xiao et al. (Apr. 2009) was guided by fundamental detection theory. If we are assuming a maximum likelihood criterion and if the additive noise $\mathbf{v}_{iL}$ is independent white Gaussian noise then a correlation detector should be used (see Poor (1994)). Since the power of the training frame and the testing frame may be significantly different a power normalization was proposed as well. The resulting similarity measure becomes

$$\sigma(\mathbf{x}_{iL}, \hat{\mathbf{s}}_n) = \frac{\mathbf{x}_{iL}^{\mathrm{T}}\, \hat{\mathbf{s}}_n}{\sqrt{\|\mathbf{x}_{iL}\|^2 - V^2} \cdot \|\hat{\mathbf{s}}_n\|}, \tag{4}$$

in which $\sqrt{\|\mathbf{x}_{iL}\|^2 - V^2}$ represents the estimated power of the underlying clean speech $s[n]$. If $\mathbf{v}_{iL}$ contains colored noise then a prewhitening filter is used before the correlation detector. With $\mathbf{h}_w$ denoting the impulse response of the prewhitening filter we obtain

$$\acute{\sigma}(\mathbf{x}_{iL}, \hat{\mathbf{s}}_n) = \frac{(\mathbf{x}_{iL} * \mathbf{h}_w)^{\mathrm{T}}\, (\hat{\mathbf{s}}_n * \mathbf{h}_w)}{\sqrt{\|\mathbf{x}_{iL} * \mathbf{h}_w\|^2 - V_w^2} \cdot \|\hat{\mathbf{s}}_n * \mathbf{h}_w\|}, \tag{5}$$

where we use $V_w^2 = \mathrm{E}\{(\mathbf{v}_n * \mathbf{h}_w)^{\mathrm{T}}(\mathbf{v}_n * \mathbf{h}_w)\}$ to denote the variance of the prewhitened noise. A discussion of the non-stationary noise case is omitted here since it is effectively using the same similarity measure as the colored noise case.
After the generation of an appropriate similarity measure between an incoming noisy frame $\mathbf{x}_{iL}$ (or $\tilde{\mathbf{x}}_{iL}$) and an inventory element $\mathbf{s}_n$ we can define an optimal intra cluster match $\hat{\mathbf{s}}^{(i,k)}$ via

$$\hat{\mathbf{s}}^{(i,k)} = \underset{\hat{\mathbf{s}}_n \in \mathbb{K}_k}{\arg\max}\; \sigma(\mathbf{x}_{iL}, \hat{\mathbf{s}}_n). \tag{6}$$

In a last step we need to resynthesize our targeted signal. First, we are replacing each frame $\mathbf{x}_{iL}$ with the inventory frame $\hat{\mathbf{s}}^{(i,k_{opt}(i))}$, i.e. $\mathbf{x}_{iL} \rightarrow \hat{\mathbf{s}}^{(i,k_{opt}(i))}$, and second, we reconcatenate the resulting frames via a *sinusoidal model expansion* similar to the one proposed by Quatieri (2002). The reconcatenation with the sinusoidal model is important to minimize phase incompatibilities at the frame boundaries.
The performance of the proposed method was evaluated by Xiao et al. (Apr. 2009) with experiments over a subset of the `CMU_ARCTIC` database from the Language Technologies Institute at Carnegie Mellon University[3]. Data processing was conducted at a sampling rate of 8 kHz

[3] The corpus is available at <http://www.festvox.org/cmu_arctic>.

and with a segment length of $N = 160$ samples and a step size of $L = 80$ samples. The targeted signal-to-noise ratio was 10 dB. The quality of the resulting denoised speech was assessed with the *Perceptual Evaluation of Speech Quality*[4] (PESQ) measure. For white noise Xiao et al. reported an improvement of up to 1.06 points on the PESQ scale. For colored noise an improvement of up to 0.87 points was reported. The performance of the presented method compared favorably with other state-of-the-art denoising methods.

3. Fast Processing Methods

As mentioned earlier, the bottleneck of the proposed denoising procedure, in terms of computational complexity, can be found in the maximization of equations (4) and (5) as expressed in equation (6). In the experiments conducted by Xiao et al. (Apr. 2009) training sets of around 1 hour in length were used. If we operate at a sampling rate of 8 kHz and with a number of $M = 50$ clusters then we can expect to have around $500 \cdot 10^3$ to $600 \cdot 10^3$ samples per cluster. For the denoising of each incoming frame we need to correlate a vector of length 160 with the entire data set contained in the cluster targeted by $k_{opt}(i)$. The resulting computational complexity per frame is therefore huge, if no fast computational procedures are involved.
For the remainder of this section we will discuss methods that can dramatically reduce the computational complexity of the maximization implied in equation (6). In a first step we are moving to a slightly simplified similarity measure since for a fixed $\mathbf{x}_{iL}$ the terms $\sqrt{\|\mathbf{x}_{iL}\|^2 - V^2}$ and $\sqrt{\|\mathbf{x}_{iL} * \mathbf{h}_w\|^2 - V_w^2}$ remain unchanged and can therefore be dropped from the computation:

$$\tilde{\sigma}(\mathbf{x}, \hat{\mathbf{s}}_n) = \frac{\mathbf{x}^{\mathrm{T}} \cdot \hat{\mathbf{s}}_n}{\|\hat{\mathbf{s}}_n\|}. \tag{7}$$

Similarity measure (5) can be modified accordingly. The respective result for equation (5) is obtained by substituting $\mathbf{x}_{iL} * \mathbf{h}_w$ and $\hat{\mathbf{s}}_n * \mathbf{h}_w$ into equation (7).
The fast computation of (7) for all frames in a given cluster (as expressed in equation (6)) is accomplished in three steps:

1. Quantization of the elements in $\mathbf{x}$ and $\hat{\mathbf{s}}_n$.
2. Computation of $\mathbf{x}^{\mathrm{T}} \cdot \hat{\mathbf{s}}_n$ with an overlap-add based convolution procedure via number theoretic transforms (NTTs).
3. Recursive computation of $\|\hat{\mathbf{s}}_n\|$ and $\tilde{\sigma}(\mathbf{x}, \hat{\mathbf{s}}_n)$.

We begin by applying a uniform scalar quantizer (see Sayood (1996)) to all elements of our (possibly preprocessed) incoming frame $\mathbf{x}$ and the elements of our inventory vectors $\hat{\mathbf{s}}_n$. The quantizer is designed to assign a unique integer between $-J$ and $+J$ ($J \in \mathbb{N}$) to every value in $\mathbf{x}$ and $\hat{\mathbf{s}}_n$. An optimal choice of J is dependent on three things: (1) the employed frame length N, (2) the statistics of our training data, and (3) the parameters of the employed number theoretic transforms. Good choices for J are discussed in section 3.3.
For simplicity of notation we assume for the remainder of this section that symbols $\mathbf{x}$ and $\hat{\mathbf{s}}_n$ and the associated signals $x[n]$ and $\hat{s}[n]$ are *quantized* versions of the original signals, i.e. all elements of these signals and vectors are integers in the range $-J \ldots + J$. It is important to emphasize that we are *not* operating with this kind of quantized data in *other* components of the proposed method, especially in the step $\mathbf{x}_{iL} \rightarrow \hat{\mathbf{s}}^{(i,k_{opt}(i))}$ during the target signal resynthesis where we want to use the original inventory data and not the quantized one.

[4] The PESQ measure, an ITU recommendation, is aiming to asses the *subjective quality* of speech. Please refer to the text by Loizou (2007) for the details.

3.1. Preliminary Computations and Notation

Before we delve into the fast computation of $\mathbf{x}^{\mathrm{T}} \cdot \hat{\mathbf{s}}_n$ it is beneficial to first briefly discuss an efficient way to compute the term $\|\hat{\mathbf{s}}_n\|$ in (7). For notational convenience we introduce the shifted inventory signal $s'[n] = \hat{s}[n - L + N - 1]$ and its square $\varsigma[n] = s'[n] \cdot s'[n]$. We can use $\varsigma[n]$ as an input to the following recursive system with output $\xi[n]$:

$$\xi[n] = \xi[n-1] + \varsigma[n] - \varsigma[n-N]. \tag{8}$$

Term $\|\hat{\mathbf{s}}_n\|$ is then obtained from $\|\hat{\mathbf{s}}_n\| = \sqrt{\xi[n]}$. The computation of $\|\hat{\mathbf{s}}_n\|$ therefore requires one multiplication, two additions[5], and one square root operation (table lookup) per sample.
A major step in simplifying the computation of $\mathbf{x}^{\mathrm{T}} \cdot \hat{\mathbf{s}}_n$ is obtained from recognizing that the inner product in (7) is equivalent to a convolution operation (indicated with symbol $*$):

$$\mathbf{x}^{\mathrm{T}} \cdot \hat{\mathbf{s}}_n = \sum_{k=0}^{N-1} [\mathbf{x}]_{N-k} \cdot \hat{s}[n-k+N-L-1] = [\mathbf{x}]_{N-k} * s'[n]. \tag{9}$$

We use the notation $[\mathbf{x}]_k$ to indicate the k^{th} element of vector $\mathbf{x}$ with $[\mathbf{x}]_k = 0$ if $k < 1$ and $k > N$. The convolution of equation (9) is further broken down by segmenting $s'[n]$ into segments of length R. The segments are zero padded to arrive at a processing block-length of K samples:

$$\mathbf{s}'_k = [\ s'[kR]\ \ s'[kR+1]\ \ \ldots\ \ s'[kR+R-1]\ \ \underbrace{0\ 0\ \ldots\ 0}_{K-R}\ ]^{\mathrm{T}}. \tag{10}$$

Similarly we are defining a time-reversed and zero padded input signal vector $\mathbf{x}'$ as:

$$\mathbf{x}' = [\ [\mathbf{x}]_N\ \ [\mathbf{x}]_{N-1}\ \ \ldots\ \ [\mathbf{x}]_2\ \ [\mathbf{x}]_1\ \ \underbrace{0\ 0\ \ldots\ 0}_{K-N}\ ]^{\mathrm{T}}. \tag{11}$$

The convolution operation can then be performed via number theoretic transforms (NNTs, see Blahut (1987)). The details of the proposed NTT operation are described in section 3.2. If we assume that we have access to the output vector $\mathbf{y}'_k = \mathrm{NTTConv}\{\mathbf{x}', \mathbf{s}'_k\}$ of the proposed K-point NTT convolution of $\mathbf{x}'$ and $\mathbf{s}'_k$ then the inner product in equation (9) becomes:

$$\mathbf{x}^{\mathrm{T}} \cdot \hat{\mathbf{s}}_n = \textstyle\sum_k [\mathbf{y}'_k]_{n+1-kR}. \tag{12}$$

The overlap and add method implied in equation (12) requires $(K - R)$ additions for each block of length K, plus the operations necessary for NTTConv.

3.2. Number Theoretic Transforms

Number theoretic transforms can be used for efficient computations of convolutions if the underlying data is, as indicated earlier, discretized or quantized (see Blahut (1987)). NTTs generally operate in *finite fields* or *Galois fields*. The order p of the field is typically a prime number, in which case all operations within the field (addition, subtraction, multiplication, and division) are executed via a *modulo-p* arithmetic (see Blahut (1987)).
Not all number theoretic transforms are necessarily well suited for the development of fast algorithms. NTTs of a certain subclass, known as *Fermat NTTs,* however, have properties that make them superior to the commonly used *fast Fourier transform* (FFT, see Proakis & Manolakis

[5] We are counting subtractions and additions as the same since they share roughly the same computational complexity.

(1996)) in computing convolutions within a fixed-point arithmetic. The advantage of such NTTs are that: (1) an NTT can be implemented in real-valued arithmetic (i.e. it does not require an underlying complex number representation), and (2) many of the multiplications required for the computation simplify to *shift* operations if the underlying processing hardware is utilizing binary number representations.

NTTs have, however, three important limitations that render their practical implementation significantly less flexible than that of the FFT: (1) the processing block length K is tied to (i.e. not independent of) the order p of the underlying number representation, (2) NTTs only exist for a very limited number of combinations of K and p, and (3) internal overflow errors during convolution computations cannot be detected and/or flagged. A general discussion of all of these problems is beyond the scope of this book chapter. The interested reader may consult the literature, especially the text by Blahut (1987), for a detailed discussion. We will address the three issues above only within the context of the proposed denoising scheme.

Out of the general set of possible combinations for K and p we found that $K = 1024$ and $p = 2^{16} + 1$ (Fermat prime) are quite well suited for the proposed algorithm. We begin by considering a general integer vector $\mathbf{v} = [\ v_1\ v_2\ \dots\ v_K\ ]^{\mathrm{T}}$ of length K. More specifically, we assume that all elements v_k of $\mathbf{v}$ are integers between $-\frac{p-1}{2}$ and $+\frac{p-1}{2}$. Furthermore, we define the following warping operation:

$$\operatorname{warp}\{v_k\} = \begin{cases} v_k & \text{if } v_k \geq 0 \\ v_k + p & \text{if } v_k < 0. \end{cases} \tag{13}$$

The notation $\tilde{\mathbf{v}} = \operatorname{warp}\{\mathbf{v}\}$ refers to an application of the warp-function on vector $\mathbf{v}$ on an element-by-element basis. We also require the following dewarping mapping:

$$\operatorname{dewarp}\{\tilde{v}_k\} = \begin{cases} \tilde{v}_k & \text{if } \tilde{v}_k \leq \frac{p-1}{2} \\ \tilde{v}_k - p & \text{if } \tilde{v}_k > \frac{p-1}{2}. \end{cases} \tag{14}$$

Again, $\mathbf{v} = \operatorname{dewarp}\{\tilde{\mathbf{v}}\}$ refers to an application of the dewarp-function on vector $\tilde{\mathbf{v}}$ on an element-by-element basis. Given the parameters $K = 1024$ and $p = 2^{16} + 1$ we arrive at the following definition for the employed NTT:

$$\tilde{\mathbf{V}} = \operatorname{NTT}\{\tilde{\mathbf{v}}\} \quad \text{such that} \quad [\tilde{\mathbf{V}}]_{k+1} = \sum_{i=0}^{1023} \omega^{ik}\, [\tilde{\mathbf{v}}]_{i+1} \bmod p \quad \text{for} \quad k = 0 \dots 1023. \tag{15}$$

Number ω must be chosen such that $\omega^{1024} \bmod p = 1$. There are a number of values ω that satisfy this conditions. Not all choices, however, are well suited for the design of a fast algorithm. It is possible to apply the *radix-two Cooley-Tukey divide-and-conquer* approach (see Blahut (1987)) to equation (15). With the special choice of $\omega = 18990$ we obtain:

$$[\tilde{\mathbf{V}}]_{32k'+k''} = \sum_{i'=0}^{31} 2^{i'k'} \left[\omega^{i'k''} \sum_{i''=0}^{31} 2^{i''k''}\, [\tilde{\mathbf{v}}]_{i'+32i''} \right] \bmod p \quad \text{for} \quad k', k'' = 0 \dots 31. \tag{16}$$

It is, therefore, possible to divide the 1024-point NTT from equation (15) into two sets of 32 sub-NTTs of length 32 each and one set of 1024 multiplications with $\omega^{i'k''}$. Furthermore, the 32-point sub-NTTs can be computed *without any multiplications* since a multiplication with a power-of-two number is equivalent to a *shift operation* if the underlying processing hardware operates on a binary number system. A detailed analysis of equation (16) reveals that equation (15) can be computed with roughly 1024 multiplications, 5120 additions, and 5120 shift operations.

An important aspect of the computations in equations (16) and (15) is the modulo reduction of order p. Given the standard approach to modulo reductions one might suspect that each reduction comes at the cost of an integer division. In the given case of $p = 2^{16} + 1$, however, it becomes possible to reduce the complexity of a modulo reduction to that of an addition, if the underlying processing hardware operates on a binary number system. To that end, we can group adjacent bits in our underlying number representation into blocks of 16. All blocks with bits higher than 16 are shifted down to line up with the least significant bit and then added block-by-block with an alternating sign. The details of the implementation are readily found in the literature (see also Blahut (1987)). Due to the cyclic nature of the $p = 2^{16} + 1$ reduction it is possible to build the require modulo operation directly into the hardware of the employed multiplier, adder, and shifting units. We, therefore, do not count modulo operations separately in our complexity analysis.

The computation of convolutions via NTTs requires us to also consider the inverse NTT. Similarly to equation (15) we define:

$$\tilde{\mathbf{v}} = \mathrm{NTT}^{-1}\{\tilde{\mathbf{V}}\} \quad \text{such that} \quad [\tilde{\mathbf{v}}]_{k+1} = K^{-1} \sum_{i=0}^{1023} \omega^{-ik} \, [\tilde{\mathbf{V}}]_{i+1} \mod p \qquad \text{for} \quad k = 0 \ldots 1023. \tag{17}$$

Note that (K^{-1}) represents the integer with the property $(K^{-1}) \cdot K \bmod p = 1$. The inverse NTT can also be computed via the *radix-two Cooley-Tukey divide-and-conquer* approach:

$$[\tilde{\mathbf{v}}]_{32k'+k''} = K^{-1} \cdot \sum_{i'=0}^{31} 2^{-i'k'} \left[\omega^{-i'k''} \sum_{i''=0}^{31} 2^{-i''k''} \, [\tilde{\mathbf{V}}]_{i'+32i''} \right] \mod p \qquad \text{for} \quad k', k'' = 0 \ldots 31. \tag{18}$$

The computational complexity of the NTT and the inverse NTT are therefore the same, except we have an additional set of 1024 multiplications with K^{-1} in the case of the inverse NTT. We will see in section 3.3 though that the scaling with K^{-1} can be omitted when we apply the inverse NTT to our proposed fast computation procedure.

We are now in the position to define the computation of $\mathbf{y}'_k$ as used in equation (12). The operation requires two warping operation, one dewarping operation, two NTTs, and one inverse NTT:

$$\mathbf{y}'_k = \mathrm{NTTConv}\{\mathbf{x}', \mathbf{s}'_k\} = \mathrm{dewarp}\{\mathrm{NTT}^{-1}\{\mathrm{NTT}\{\mathrm{warp}\{\mathbf{x}'\}\} \odot \mathrm{NTT}\{\mathrm{warp}\{\mathbf{s}'_k\}\}\}\}, \tag{19}$$

in which $\odot$ denotes element-by-element-wise vector multiplication.

3.3. Complexity Analysis and Perfomance

A successful implementation of the fast algorithm proposed in the previous two sections requires a careful definition of our quantization granularity[6] J. If J is too big then we are likely to receive too many (undetectable) overflow errors in the NTT based convolution operation.

[6] Note that the effective number of quantization levels for our data is given by $2J + 1$.

The resulting intra cluster frame matching becomes unreliable and the perceptual quality of the proposed denoising method suffers. If we pick J too small then the effective quantization granularity of our data becomes too coarse. Again, the resulting intra cluster frame matching becomes unreliable and the perceptual quality is reduced.
In experiments over the same data set that was used in the original performance analysis by Xiao et al. (Aug. 2008) we found that the error count in the intra cluster frame matching procedure remained relatively unaffected by the number of employed quantization levels if the number did not drop significantly below 60, i.e. $2J + 1 \geq 60$. Similar experiments revealed that we receive virtually no overflow errors in our procedure[7] if $J \leq 35$. A recommended range for J is therefor between 30 and 35.
To maximize the efficiency of the proposed NTT based convolution it is best to pick $K = N + R - 1$. With a processing block-length K of 1024 and a frame length N of 160 we obtain $R = 865$.
To obtain reasonably normalized numbers for the computational complexity of different solution approaches for equation (6) we decided to reference all operation counts to an equivalent count for each inventory sample. A direct, brute force, computation of equation (6) requires 320 multiplications/sample and 318 additions/sample. We technically also require one division/sample and one square-root-operation/sample. The division and the square root, however, are a part of all considered algorithms and are therefore omitted in the overall counts.
For comparison we consider the proposed fast convolution approach with a conventional *radix-two fast Fourier transform* (FFT) instead of the proposed NTT. A 1024-point FFT requires 9216 complex multiplications and 10240 complex additions[8] (see Blahut (1987)). Each complex multiplication can be evaluated with 3 real multiplications and 5 real additions. We, therefore, obtain 27648 real multiplications and 56320 real additions. The disadvantage of a complex arithmetic of the FFT is partially alleviated by the fact that we can typically process two FFTs with real imputs with a single FFT with complex inputs (see Proakis & Manolakis (1996)). Operations are consequently cut in half and we obtain as a final count for a 1024-point FFT 13824 (real) multiplications and 28160 (real) additions. The FFT equivalent of equation (19) can be evaluated on-line with one 1024-point FFT, one 1024-point inverse FFT and 1024 complex multiplications. Note that we only need one FFT to compute (19) on-line since the corresponding FFTs of our inventory $\hat{s}[n]$ can be precomputed off-line. Furthermore, we do not need to consider the additional scaling factor of $\frac{1}{K}$ in the inverse FFT since the scaling becomes immaterial in the subsequent maximum search. Furthermore, we receive an additional number of $K - R = N - 1 = 1023$ additions due to the overlap and add procedure from equation (12).
In summary, we require $2 \times 13824 + 3 \times 1024 = 30720$ multiplications and $2 \times 28160 + 5 \times 1024 + 1023 = 62463$ additions to compute the required 1024-point convolution with an FFT based approach. The convolution operation has to be repeated every $R = 865$ samples. On a per-sample count we obtain 35.52 multiplications/sample and 72.22 additions/sample. Technically we need to also add in the one multiplication/sample and the two additions/sample for the separate computation of $\|\hat{\mathbf{s}}_n\|$.

[7] Assuming $N = 160$, $K = 1024$, and $p = 2^{16} + 1$.
[8] The complexity analysis presented here may slightly differ from complexity computations from other sources. The main differences in computation counts are usually due to differences in how trivial multiplications are considered. We decided to include the count of trivial multiplications for the FFT as well as the NTT.

The computation of a 1024-point NTT after section 3.2 requires 1024 multiplications, 5120 additions, and 5120 shift operations. Similarly to the computations for the FFT we require $2 \times 1024 + 1024 = 3072$ multiplications, $2 \times 5120 + 1023 = 11263$ additions, and $2 \times 5120 = 10240$ shift operations to compute the required 1024-point convolution with an NTT based approach. Again, the convolution operation has to be repeated every $R = 865$ samples. On a per-sample count we obtain 3.56 multiplications/sample, 13.03 additions/sample, and 11.84 shifts/sample. Considering also the one multiplication/sample and the two additions/sample for the separate computation of $\|\hat{\mathbf{s}}_n\|$ we obtain a total tally of 4.56 multiplications/sample, 15.03 additions/sample, and 11.84 shifts/sample for the proposed approach.

4. Conclusions

We presented a *fast algorithm* for the correlation computations that are required for the inventory based speech enhancement method proposed by Xiao et al. (Apr. 2009). The correlation computations are used in the inventory unit selection scheme of the enhancement procedure. They present a significant computational bottleneck for this method. The computational complexity of the inventory unit selection scheme would dominate the overall processing requirement of the method by an order of magnitude if no fast algorithms were employed.
The fast computation procedure proposed in this chapter is able to dramatically reduce the computational complexity of the proposed method without significantly affecting its enhancement performance. The number of multiplications per inventory sample required for the processing can be reduced from around 36.52 for a conventional FFT based method down to around 4.56 for the proposed NTT based method. The proposed approach is thus significantly faster than conventional computation methods.

5. References

Blahut, R. E. (1987). *Fast Algorithms for Digital Signal Processing*, Addison-Wesley.

Deller, J. R., Proakis, J. G. & Hansen, J. H. (1993). *Discrete-Time Processing of Speech Signals*, Macmillan, New York.

Ealey, D., Kelleher, H. & Pearce, D. (2001). Harmonic tunnelling: tracking non-stationary noises during speech, *Proceedings of EUROSPEECH* pp. 437–440.

Ephraim, Y. & Malah, D. (1984). Speech enhancement using a minimum mean square error short-time spectral amplitude estimator, *IEEE Transactions on Acoustics, Speech, and Signal Processing* **32**: 1109–1121.

Hendriks, R. C., Heusdens, R. & Jensen, J. (2007). An MMSE estimator for speech enhancement under a combined stochastic/deterministic speech model, *IEEE Transactions on Audio, Speech and Language Processing* **15**(2): 406–415.

Hu, Y. & Loizou, P. C. (2004). Speech enhancement based on wavelet thresholding the multitaper spectrum, *IEEE Transactions on Speech and Audio Processing* **12**(1): 59–67.

Loizou, P. C. (2007). *Speech Enhancement, Theory and Practice*, CRC-Press.

McAulay, R. J. & Malpass, M. L. (1980). Speech enhancement using a soft-decision noise suppression filter, *IEEE Transactions on Acoustics, Speech, and Signal Processing* **ASSP-28**(2): 137–145.

Mouchtaris, A., Van der Spiegel, J., Mueller, P. & Tsakalides, P. (2007). A spectral conversion approach to single-channel speech enhancement, *IEEE Transactions on Audio, Speech and Language Processing* **15**(4): 1280–1193.

O'Shaughnessy, D. (2007). Modern methods of speech synthesis, *IEEE Circuits and Systems Magazine* **7**(3): 6–23.

Poor, H. V. (1994). *An Introduction to Signal Detection and Estimation*, Springer-Verlag.

Proakis, J. G. & Manolakis, D. G. (1996). *Digital Signal Processing – Principles, Algorithms, and Applications*, 3rd edn, Prentice Hall, Upper Saddle River, New Jersey 07458.

Quatieri, T. F. (2002). *Discrete-Time Speech Signal Processing: Principles and Practice*, Prentice Hall.

Sayood, K. (1996). *Introduction to Data Compression*, Morgan Kaufman.

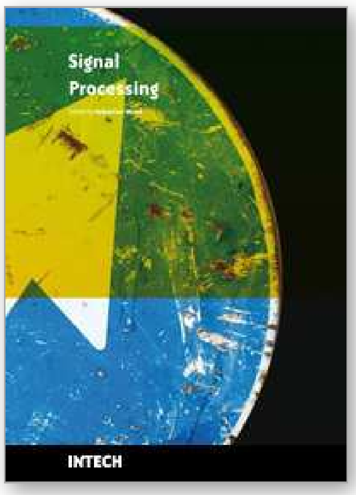
Srinivasan, S., Samuelsson, J. & Kleijn, W. B. (2006). Codebook driven short-term predictor parameter estimation for speech enhancement, *IEEE Transactions on Audio, Speech, and Language Processing* **14**(1): 163–176.

Xiao, X., Lee, P. & Nickel, R. M. (Apr. 2009). Inventory based speech enhancement for speaker dedicated speech communication systems, *Proceedings of ICASSP*, Taipei, Taiwan, pp. 3877–3880.

Xiao, X., Lee, P. & Nickel, R. M. (Aug. 2008). Inventory based speech denoising with hidden Markov models, *Proceedings of EUSIPCO*, Lausanne, Switzerland.

Zavarehei, E., Vaseghi, S. & Yan, Q. (2007). Noisy speech enhancement using harmonic-noise model and codebook-based post-processing, *IEEE Transactions on Audio, Speech, and Language Processing* **15**(4): 1194–1203.

Zhao, D. Y. & Kleijn, W. B. (2007). HMM-based gain modeling for enhancement of speech in noise, *IEEE Transactions on Audio, Speech, and Language Processing* **15**(3): 882–892.



Signal Processing
Edited by Sebastian Miron

ISBN 978-953-7619-91-6
Hard cover, 528 pages
Publisher InTech
Published online 01, March, 2010
Published in print edition March, 2010

This book intends to provide highlights of the current research in signal processing area and to offer a snapshot of the recent advances in this field. This work is mainly destined to researchers in the signal processing related areas but it is also accessible to anyone with a scientific background desiring to have an up-to-date overview of this domain. The twenty-five chapters present methodological advances and recent applications of signal processing algorithms in various domains as telecommunications, array processing, biology, cryptography, image and speech processing. The methodologies illustrated in this book, such as sparse signal recovery, are hot topics in the signal processing community at this moment. The editor would like to thank all the authors for their excellent contributions in different areas of signal processing and hopes that this book will be of valuable help to the readers.