SIP (2019), vol. 8, e4, page 1 of 20 © The Authors, 2019.

doi:10.1017/ATSIP.2018.23

ORIGINAL PAPER

Development of a computationally efficient voice conversion system on mobile phones

SHUHUA GAO[1,2], XIAOLING WU[1,2], CHENG XIANG[1] AND DONGYAN HUANG[2]

Voice conversion aims to change a source speaker's voice to make it sound like the one of a target speaker while preserving linguistic information. Despite the rapid advance of voice conversion algorithms in the last decade, most of them are still too complicated to be accessible to the public. With the popularity of mobile devices especially smart phones, mobile voice conversion applications are highly desirable such that everyone can enjoy the pleasure of high-quality voice mimicry and people with speech disorders can also potentially benefit from it. Due to the limited computing resources on mobile phones, the major concern is the time efficiency of such a mobile application to guarantee positive user experience. In this paper, we detail the development of a mobile voice conversion system based on the Gaussian mixture model (GMM) and the weighted frequency warping methods. We attempt to boost the computational efficiency by making the best of hardware characteristics of today's mobile phones, such as parallel computing on multiple cores and the advanced vectorization support. Experimental evaluation results indicate that our system can achieve acceptable voice conversion performance while the conversion time for a five-second sentence only takes slightly more than one second on iPhone 7.

Keywords: Voice conversion, GMM, Mobile application, Parallel computing, Weighted frequency warping

Received 1 July 2018; Revised 8 November 2018

I. INTRODUCTION

Voice conversion refers to the modification of the voice of a speaker, called the *source speaker*, to make it sound as if it was uttered by another speaker, called the *target speaker*, while leaving the linguistic content unchanged. Therefore, the voice characteristics of the source speaker must be identified and transformed by a voice conversion system to mimic those of the target speaker, without changing the message transmitted in the speech.

Voice conversion has great potential in developing various industrial applications, including not only integration into a text-to-speech synthesis systems (TTS) to customize the voice as we like [1], but also serving a role in rehabilitation medicine, entertainment, and education areas. For example, by transforming the vowels of a speaker with *dysarthria*, a speech motor disorder, into the vowel space of a non-dysarthria speaker, the speech intelligibility has been improved significantly [2]. Besides, voice conversion technologies have also been used to produce varied multi-singer voices by applying a conversion model to a single-singer database [3], to synthesize emotional speech from standard reading (neutral) speech [4] or transform the emotion conveyed in a speech to another one [5], and to generate prosodically corrected versions of foreign language learners' utterances in the computer-assisted pronunciation training [6, 7]. On the contrary, voice conversion may also present a threat to speaker verification and has thereby forced the enhancement of anti-spoofing capacities of typical speaker verification systems [8]. All in all, it deserves our efforts to study voice conversion for both scientific research and industrial application purposes.

Roughly speaking, human speech can be decomposed into three components: language content, spectral pattern, and prosody [8]. The last two factors form the focus of current voice conversion studies located at two levels: the supra-segmental level, which includes prosodic features such as the fundamental frequency contour, the duration of words, phonemes, timing, rhythm, and intensity levels etc.; and the segmental level including the average pitch intensity, the frequency response of the vocal tract, and the glottal source characteristics. In this sense, voice conversion aims to map the spectral and prosodic features of the source speaker to the ones of the target speaker in order to modify voice individuality. Such a feature mapping task is commonly formulated as a supervized learning problem in the literature, which requires a certain amount of speech data available from both the source speaker and the target speaker for training. The key difference among various

[1]Department of Electrical & Computer Engineering, National University of Singapore, Singapore
[2]Human Language Technology Department, Institute for Infocomm Research, A*STAR, Singapore

Corresponding author:
Dongyan Huang
Email: huang@i2r.a-star.edu.sg

studies lies in the specific mapping models adopted, mainly including codebook-based, statistical and artificial neural network-based methods, which have been reviewed comprehensively in a recent survey [9]. In the following, we only discuss some representative approaches briefly to explain the rationality of our methodology.

A typical voice conversion system requires two types of models, i.e., a speech model and a conversion model. After extensive investigations, it has been found that the three most relevant speech features regarding speaker individuality are the average spectrum, the formants, and the average pitch level [6]. Consequently, most voice conversion systems attempt to transform short-time spectral envelopes and prosodic characteristics including the pitch value, duration, and intensity [9]. Thus, before voice conversion, a suitable speech model must be adopted to analyze the input speech signals so as to extract the relevant speech features for subsequent modifications. Additionally, a good speech model should be able to reconstruct the speech signal from model parameters with high quality, a capacity we need to synthesize speech for the target speaker after conversion. Usually, the speech model is built frame-by-frame while each frame represents a short-time segment (typically $<$20 ms).

Popular speech models include the STRAIGHT model [10], the pitch-synchronous overlap-add (PSOLA) model [11], and the harmonic plus noise model (HNM) [12], among others. STRAIGHT belongs to the filter model family, which assumes the speech is produced by filtering an excitation signal with a vocal tract filter independent of the excitation. To alleviate the interference between signal periodicity and the spectral envelope, STRAIGHT performs a pitch-adaptive spectral analysis, which can preserve the details of time-frequency surfaces while almost perfectly removing minor structures caused by signal periodicity. On the other hand, signal-based methods such as PSOLA and HNM do not model speech as a combination of a source signal and a spectral envelope filter, thereby avoiding restrictive assumptions like the independence between the source signal and the filter, which can usually lead to higher quality of speech synthesis. Specifically, the PSOLA methods can modify the speech signal either in the frequency domain (FD-PSOLA), or directly in the time domain (TD-PSOLA). Particularly, TD-PSOLA decomposes the speech waveform into a stream of short-time analysis signals set at a pitch synchronous rate, which is one of the most popular and simplest methods for high-quality prosodic modification. Nonetheless, in the aspect of voice conversion, models based on the sinusoidal decomposition of speech are more desirable, because such models allow flexible spectral and prosodic modifications by manipulating model parameters. As a typical representative, HNM is one of the most widely used models for speech synthesis and modification [12]. HNM decomposes the speech signal into a deterministic part as a sum of sinusoids with frequencies relevant to pitch and a stochastic part obtained by filtering a white Gaussian noise. A comparison study in [13] shows that HNM gives better overall performance than TD-PSOLA, and thereby a speech model based on such harmonic plus stochastic decomposition is adopted in our system.

Derived from the above speech modeling and analysis step, the output features may be directly used or further processed for subsequent feature mapping [9]. As aforementioned, in current voice conversion, most methods assume frame-by-frame processing and thus depend mainly on local speech features present in short-time segments. Common spectral features include Mel-cepstral coefficients (MCCs), linear predictive cepstral coefficients(LPCCs), line spectrum frequency (LSF) and formant frequencies, and bandwidths [8, 9]. Regarding the prosodic features, we usually only consider the f_0 and the duration patterns. Typical voice conversion systems only perform simple modifications of prosodic features, e.g., normalizing the global mean and variance of log-scaled f_0 values [8, 14], because prosody is a supra-segmental characteristic not conveyed on segmental basis but through larger units, which is more challenging [15]. Consequently, most researches focus on mapping for the spectral features. Mathematically speaking, voice conversion is to learn a mapping function $f(\cdot)$ from the source speech feature $\boldsymbol{x}$ to the target speech feature $\boldsymbol{y}$ using the training corpus and then apply this function to a new unseen source speech sample for conversion at runtime.

Starting from the seminal work based on vector quantization (VQ) and mapping codebooks developed by Abe et al. in 1988 [16], many methods for spectral feature mapping have been proposed in the literature. Though the codebook mapping-based methods are simple and computationally efficient, they performed poorly due to the generation of discontinuous feature sequences [9, 15]. At present, more popular methods are mainly classified into the following categories: (1) mixture of linear mappings, like Gaussian mixture models (GMM) [13, 17, 18] and hidden Markov models (HMM) [19, 20]; (2) a single nonlinear mapping model, including support vector machine regression [21], artificial neural networks (ANN) [22, 23], and the more recent deep learning approaches [24, 25]; (3) frequency warping-based mapping approaches such as weighted frequency warping [26], dynamic frequency warping [27], and bilinear frequency warping [28]; and (4) nonparametric mapping methods like the exemplar-based method [29, 30], Gaussian Process regression [31], and the K-histogram approach [32]. In GMM-based methods, the distribution of the source (target) spectral feature vectors are modeled with a group of multivariate normal distributions, and the conversion function typically adopts a linear form and thus can be solved with least squares directly [13, 15]. Another way to handle the mapping in GMM is to fuse the source and target feature vectors together to build a GMM for the augmented vector $\boldsymbol{z} = [\boldsymbol{x}^T, \ \boldsymbol{y}^T]^T$, called a joint GMM, and the mapped target vector during conversion can be acquired using the mean vectors and the covariance matrices we have obtained from training [33]. In ANN-based approaches, both shallow and deep networks have been investigated to map the features from the source to the target nonlinearly, which may outperform GMM-based methods due to the extremely flexible structure and the prominent nonlinear fitting ability especially when a deep neural network (DNN) is used [23, 34]. Nevertheless, the network design and training requires plenty of expertise since the learning process can easily get stuck in local minima and DNN generally asks for a large amount of training data [35]. It is known that statistical methods like GMM tend to generate over-smoothed speech of degraded quality though they can convert speaker identity well [27, 30]. By contrast, frequency warping aims to map the frequency axis of the source speaker's spectrum to that of the target speaker by warping source spectral envelope, thereby retaining more spectral details and generally producing speech of higher quality [30, 36]. Despite the high-quality score, the similarity between converted and targeted voices, i.e., the identity conversion performance, is not satisfactory in frequency warping, because the spectral shape was not completely modified [26]. An alternative method to generate high-quality speech is the nonparametric exemplar-based voice conversion. An *exemplar* is defined as a segment of speech spectrogram spanning multiple frames, while the set of linear combination weights compose an activation vector. To avoid the over-smoothing effect, the activation vector is required to be sparse, which can be estimated by nonnegative matrix factorization (NMF) with sparse constraint or nonnegative matrix deconvolution [29]. This framework has been further extended to include spectral compression factor and a residual compensation technique [30]. Experiment results on the VOICES database demonstrate the superiority of the exemplar-based method on both converted speech similarity and quality. However, this approach has the disadvantage of high computational cost, which renders it currently unsuitable for mobile applications. Arguably, GMM is still one of the most successful and most widely used models in practical voice conversion systems [8, 9, 15, 37]. In our voice conversion system designed for smart phones, we adopted the weighted frequency warping method, which combines GMM with frequency warping to achieve better balance between speech quality and similarity [26].

In recent years, mobile devices, especially smart phones and tablets, have prevailed in our daily life. A report shows that time spent on mobile devices grew 117% between 2014 and 2015, while overall mobile application (app) usage has increased on average 58% year-over-year [38]. However, though there are many algorithms proposed and verified with specific datasets in literature, as listed above, to the best of our knowledge, there are still no voice conversion systems using such advanced methods that can work on mobile devices, particularly on mobile phones. Moreover, through a search on Apple store and Google play for present applications labeled with *voice change/conversion*, we found that most of the existing voice change applications merely attempt to simply modify some of the spectral or prosodic features with no specific target but only to generate robot-like voices for fun, such as *VoiceLab* and *Voice Changer*. Only few applications are intentionally developed to mimic another predefined person's voice, such as *Trump Voice Changer*, which can utter the input text in a Trump-like voice, and the *Celebrity Voice Changer Lite* app, which provides a fixed list of celebrities as your voice conversion targets. Even so, according to user reviews, these mobile applications (apps) exhibit obvious limitations, e.g., the produced utterance usually sounds unnatural, dissimilar, and short of intelligibility. Besides, some apps need an Internet connection to access online servers. More importantly, no apps support customization of target speakers but only provide fixed targets, missing the key features of a general voice conversion system. Therefore, this motivates us to develop a *full-fledged*, *offline*, and *real-time* voice conversion system on mobile phones by the efficient implementation of a GMM-based algorithm, which involves the best utilization of hardware vectorization instructions, parallel computing on multiple cores, and good design of the software architecture. An iOS application called *Voichap* is developed to demonstrate the feasibility of full-featured real-time voice conversion on iPhone devices.

This paper is organized as follows. In Section II, we briefly introduce the underlying algorithm framework for voice conversion in our system. In Section III, we describe the major principles and techniques for efficient implementation of the core algorithms on mobile phones. The subsequent Section IV presents the software architecture and the development of a practical mobile application for the iOS platform. Section V demonstrates the iOS application and afterward the effectiveness and efficiency of this application are evaluated experimentally. In Section VI, we give a brief discussion regarding deep learning-based approaches. Finally, we conclude the paper in Section VII.

II. VOICE CONVERSION FRAMEWORK

As mentioned above, we have reviewed, analyzed, and compared the various voice conversion methods and finally chosen a technique called Weighted Frequency Warping (WFW), which combines the outstanding conversion capacity of GMM-based approaches and the high quality of frequency-warping based transformations, proposed by Daniel Erro et al. in [26, 33, 36]. We adopted this method mainly due to its good performance yet still a relatively low computational burden since we must take the limited computational resources of mobile devices into consideration. For completeness purpose of this paper, the overview of the voice conversion framework is first illustrated and then the individual components of this framework are briefly described in this section.

A) Overview of the voice conversion system

An overview of a typical voice conversion system, including both the training phase and the conversion phase is shown in Fig. 1. Overall, in the training phase a conversion function $f(\cdot)$ is learned to map the source feature vector $\boldsymbol{x}$ to the target feature vector $\boldsymbol{y}$. Then, during the conversion phase, the feature vector $\boldsymbol{x}$ of a new source utterance is

https://doi.org/10.1017/ATSIP.2018.23 Published online by Cambridge University Press

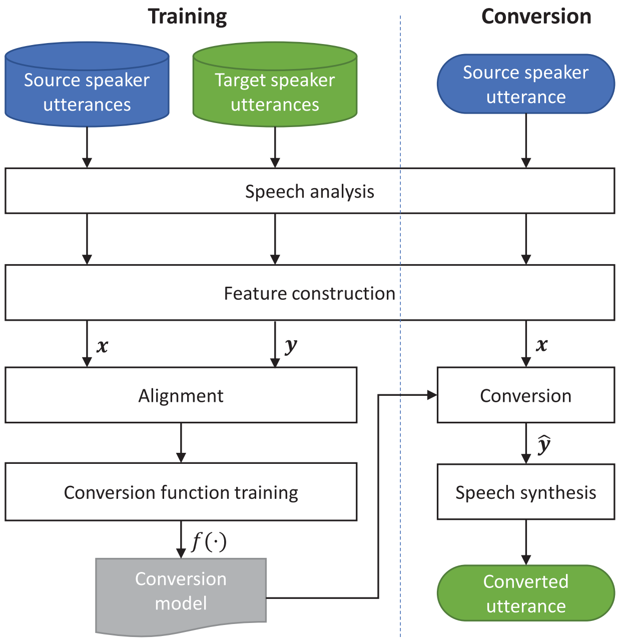


Fig. 1. General architecture of a typical voice conversion system.

built and subsequently transformed by $\hat{y} = f(x)$, which are finally used to synthesize the converted utterance in the target speaker's voice. In machine learning terminology, such voice conversion is a regression problem, but it is generally a challenging process due to that fact that speech quality and converted-to-target conversion similarity are usually two contradictory goals and a proper balance must be achieved [15]. In the subsections below, corresponding to the blocks in Fig. 1, we describe speech modeling for speech analysis, feature construction, and conversion function learning separately.

B) Deterministic plus stochastic model

In a voice conversion system, a speech model should first be good for synthesis purposes, i.e., when a speech signal is reconstructed from the model parameters, it should preserve fidelity to be perceptually indistinguishable from the original one. The aforementioned TD-PSOLA is a powerful method for speech synthesis [11]. However, it is not suitable for voice conversion because it assumes no model for the speech signal and consequently no spectral manipulation can be easily applied. Therefore, our system uses the deterministic plus stochastic model based on the sinusoidal decomposition of speech [39]. Similar to the classic HNM [12], this model represents the speech signal s as a sum of a set of sinusoids with time-varying parameters and a noise-like component denoting the residual

$$s[n] = \sum_{j=1}^{J} A_j[n] cos(\theta_j[n]) + e[n], \qquad (1)$$

where the deterministic part appears only in the voiced fragments and the stochastic part contains the remaining non-sinusoidal signal components, such as friction and breathing noise. Though the model parameters are time-varying globally, they can be considered stable within short intervals, making it reasonable and easier to analyze the signal locally by frames, where each frame corresponds to a constant number of speech samples, say N, in a time interval of 10 ms in our case.

After the amplitudes and phases are detected [40] for each measurement point k corresponding to the time instant kN, $k \geq 1$, the deterministic waveform is interpolated at every time instant by

$$A_j[kN+m] = A_j^{(k)} + \frac{A_j^{(k+1)} - A_j^{(k)}}{N} m, \qquad (2)$$

for $m = 0, 1, \ldots, N-1$, where $A_j^{(k)}$ represents the amplitude of the jth harmonic at point k. The phases and the frequencies are interpolated together by a 3rd order polynomial as follows,

$$\theta_j[kN+m] = am^3 + bm^2 + cm + d, \qquad (3)$$

where the parameters a, b, c, d are chosen in an optimal manner [41]. Now with the sinusoids available, we can obtain the complete deterministic part $d[n]$ according to equation (1), and the stochastic component $e[n]$ can be subsequently isolated as follows,

$$\begin{aligned} d[n] &= \sum_{j=1}^{J^{(k)}} A_j[n] \cos(\theta_j[n]), \\ e[n] &= s[n] - d[n], \end{aligned} \qquad (4)$$

where $J^{(k)}$ is the number of harmonics in the kth frame.

The remaining work is to analyze the magnitude spectral shape of the residual with the linear predicative coding (LPC) method. After we acquire the deterministic plus stochastic model (1), prosodies including the duration and pitch can be modified straightly, which are detailed in [41].

To reconstruct the signal from the measured parameters, the overlap-add (OLA) technique can be used to rebuild the deterministic part, where a frame of $2N$ samples is built at each measurement point k, described by

$$\begin{aligned} d[kN+m] = \sum_{j=1}^{J^{(k)}} \Bigg(& A_j^{(k)} \cos\left(w_j^{(k)} m + \phi_j^{(k)}\right) \frac{N-m}{N} \\ & + A_j^{(k+1)} \cos\left(w_j^{(k+1)}(m-N) + \phi_j^{(k+1)}\right) \frac{m}{N} \Bigg). \end{aligned} \qquad (5)$$

Next, the N-length frames of white Gaussian noise are shaped in the frequency domain by previously calculated LPC filters to generate the stochastic component. By this means, the speech signal can be successfully synthesized from model parameters. Experimental results show that the system output is perceptually indistinguishable from the original speech [41].

https://doi.org/10.1017/ATSIP.2018.23 Published online by Cambridge University Press

C) Line spectral frequencies feature construction

Though we have built a harmonic plus stochastic model, it should be noted that converting voices directly from these model parameters is quite difficult, because the amplitudes and the phases do not provide a suitable parameterization of the harmonic spectral envelope for the purpose of voice conversion. This is mainly caused by the fact that the number of harmonics is variable and generally high, which makes conversion quite complicated. Thus, as shown in Fig. 1, a further feature construction step is needed to allow better representation of speech for conversion purpose. In this study, the line spectral frequencies (LSF) coefficients, which have better quantization and interpolation properties [42], are used as the features to estimate the parameters $\{\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}$ of the Gaussian mixture model, which is introduced in the next subsection .

Given a pth order all-pole representation of the spectrum, $1/A(z)$, the LSF coefficients are the roots of the $(p+1)$th order polynomials given by

$$\begin{aligned} P(z) &= A(z) + z^{-(p+1)}A(z^{-1}), \\ Q(z) &= A(z) - z^{-(p+1)}A(z^{-1}), \end{aligned} \tag{6}$$

where P is a palindromic polynomial and Q an anti-palindromic polynomial. Note that the roots of $P(z)$ and $Q(z)$ are located in symmetric pairs on the unit circle, which means they can be completely characterized by the frequencies corresponding to the roots locations and only $p/2$ frequencies need to be stored for each polynomial. Thus, the LSF feature has p attributes in total.

Regarding practical implementation, the all-pole modeling (DAP) iterative technique is used since it leads to less distortion and thus provides better perceptual quality than its predecessors [43]. In determining the optimal order for the harmonic all-pole filters, a trade-off must be made because high-order filters provide higher resolution and higher quality, while low-order filters can be converted more reliably. Through trial and error, $p = 14$th order all-pole filters are found to provide the best results.

D) Voice conversion via weighted frequency warping

From the given training set, i.e., a parallel corpus including the source speaker's and the target speaker's utterances, the final LSF feature vector can be acquired as $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively. After time alignment, it is common to use a GMM to represent the feature distribution. We can use an augmented vector $\boldsymbol{z} = [\boldsymbol{x}^T, \boldsymbol{y}^T]^T$ to build a joint GMM with m components,

$$p(\boldsymbol{z}) = \sum_{i=1}^{m} \alpha_i \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \tag{7}$$

where $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are, respectively, the mean vector and the covariance matrix for the ith Gaussian component and $\{\alpha_i\}$ are positive adding up to 1. $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ can be partitioned into a block-wise form corresponding to $\boldsymbol{x}$ and $\boldsymbol{y}$ given by

$$\boldsymbol{\mu}_i = \begin{bmatrix} \boldsymbol{\mu}_i^x \\ \boldsymbol{\mu}_i^y \end{bmatrix}, \quad \boldsymbol{\Sigma}_i = \begin{bmatrix} \boldsymbol{\Sigma}_i^{xx} & \boldsymbol{\Sigma}_i^{xy} \\ \boldsymbol{\Sigma}_i^{yx} & \boldsymbol{\Sigma}_i^{yy} \end{bmatrix}, \tag{8}$$

which can be estimated from data using the Expectation-Maximization (EM) technique.

During conversion, for a source input vector $\boldsymbol{x}_t$ from the frame t, the conditional distribution of $\boldsymbol{y}_t$ given $\boldsymbol{x}_t$ is again modeled with mixed Gaussian distribution, and we can use the mean vector as the predicted target output vector $\hat{\boldsymbol{y}}_t$. Thus, the classic conversion function of GMM [15] is formulated as follows,

$$F(\boldsymbol{x}_t) = \sum_{i=1}^{m} \omega_i(\boldsymbol{x}_t)\Big[\boldsymbol{\mu}_i^y + \boldsymbol{\Sigma}_i^{yx}(\boldsymbol{\Sigma}_i^{xx})^{-1}(\boldsymbol{x}_t - \boldsymbol{\mu}_i^x)\Big] \tag{9}$$

where $\omega_i(\boldsymbol{x}_t)$ is the posterior probability that the given LSF vector $\boldsymbol{x}_t$ belongs to the ith Gaussian, defined by

$$\omega_i(\boldsymbol{x}_t) = \frac{\alpha_i \mathcal{N}(\boldsymbol{x}_t; \boldsymbol{\mu}_i^x, \boldsymbol{\Sigma}_i^{xx})}{\sum_{j=1}^{m} \alpha_j \mathcal{N}(\boldsymbol{x}_t; \boldsymbol{\mu}_j^x, \boldsymbol{\Sigma}_j^{xx})}. \tag{10}$$

Though the traditional GMM method described above can achieve good similarity for voice conversion, the quality of the converted speech is still unsatisfactory mainly due to the over-smoothing effect [15]. By contrast, frequency-warping-based methods can avoid significant loss of speech quality since the degree of modification is limited [35]. Therefore, we adopted the method originally proposed in [36], which aims to obtain high converted-to-target similarity while preserving the speech quality, by combining GMM and frequency warping. The main inspiration is that the mean LSF vectors of each acoustic class corresponding to each Gaussian component in GMM, $\boldsymbol{\mu}_i^x$ and $\boldsymbol{\mu}_i^y$, have a very similar formant structure. Following this observation, a piecewise linear frequency-warping function $W_i(f)$ is developed using the position of these formants for each acoustic class, and the frequency-warping function for the complete frame including m acoustic classes is obtained by

$$W(f) = \sum_{i=1}^{m} w_i(\boldsymbol{x}) W_i(f). \tag{11}$$

With the established $W(f)$, supposing $A(f)$ and $\theta(f)$ are the magnitude and phase estimators of the current frame's spectrum, spectral modifications can be made by warping the source envelopes as follows,

$$A_w(f) = A(W^{-1}(f)), \quad \theta_w(f) = \theta(W^{-1}(f)). \tag{12}$$

Even after we obtain the warped spectrum of the current frame $S_w(f)$ as above, the energy distribution still differs from the one of the actual target voice, because the intensity, bandwidth, and spectral tilt remain almost unchanged. The GMM transformation function (9) is used here to obtain a

https://doi.org/10.1017/ATSIP.2018.23 Published online by Cambridge University Press

new version of the target spectrum $S_g(f)$ from the transformed LSF vector $F(\boldsymbol{x})$. The final converted spectrum for the current frame is obtained by

$$S'(f) = G(f)S_w(f), \tag{13}$$

where the energy correction filter $G(f)$ is given by

$$G(f) = \left| \frac{S_g(f)}{S_w(f)} \right| * B(f), \tag{14}$$

which uses a smoothing window function $B(f)$ to do convolution (the * operator). The $B(f)$ function can control the balance between similarity and quality of the converted speech by adjusting its shape. A triangular smoothing function is commonly used in practice [26].

To change the magnitude of the formants, the all-pole envelope of the converted $\hat{\boldsymbol{y}} = F(\boldsymbol{x})$ can be acquired and the energy at certain bands of interest can be measured. The energy of the converted speech frame in each band is simply corrected with constant multiplicative factors.

An important prosodic feature, the fundamental frequency f_0, is modified by the simple but popular mean-variance global transformation [35]. Since f_0 follows a log-normal distribution, the mean μ_{f_0} and the variance σ_{f_0} of log f_0 can be computed during training. Then, f_0 of the converted speech signal can be linearly scaled by

$$\log f_0^{(c)} = \mu_{f_0}^{(t)} + \frac{\sigma_{f_0}^{(t)}}{\sigma_{f_0}^{(s)}} \left(\log f_0^{(s)} - \mu_{f_0}^{(s)} \right), \tag{15}$$

where the superscripts (s) and (t) represent the source and target respectively, and $f_0^{(c)}$ is the fundamental frequency of the converted speech.

As for the stochastic component in equation (1), its conversion is known to be not as relevant as the harmonic/deterministic conversion [13]. Nevertheless, it is better to predict the stochastic component for the target speaker using the vocal tract LSF parameters in voiced frames, as detailed in [36]. At this point, we have finished the description of the underlying voice conversion theory for our system. In the following section, we will detail the highly efficient implementation of such algorithms.

III. EFFICIENT IMPLEMENTATION OF CORE ALGORITHMS

Compared with laptops and PCs (personal desktop computers), mobile devices, e.g., smart phones, are generally characterized by quite limited memory, power and computational resources. Nonetheless, the main objective of our study is to develop a full-fledged voice conversion system on mobile phones without server-side support. Thus, the key challenge is to implement the conversion algorithm efficiently enough such that the time required by voice conversion, especially the conversion phase, is acceptable for daily use. To address this problem, the following two points should be mainly taken into account:

- *Algorithm efficiency*. The methods chosen for speech modeling, synthesis, and conversion are expected to be time-saving themselves. In Section II, we have introduced the involved algorithms for our voice conversion system, which are adopted partially due to their low computational burden yet still acceptable performance. For instance, the previously described deterministic plus stochastic model for speech modeling is indeed pitch-asynchronous, which can greatly simplify the analysis since the accurate separation of the signal periods is not necessary. More importantly, this conversion algorithm can even work well with a small number of training samples, e.g., tens of sentences.
- *Implementation efficiency.* This is the task on which we put most emphasis during this study. Generally speaking, the hardware is always faster than software. Thus, it is critical to make full use of the advanced hardware characteristics present on common mobile phones, including support for vectorized calculation and multi-core parallel computation. Besides, the choice of a proper programming language and smart manipulations of matrices also play an important role in acceleration.

In this section, we detail the key implementation aspects regarding computational efficiency.

A) Multi-core parallel computing

Today, it is almost standard practice for smart phones to ship a multi-core CPU, even for low-end smart phones. As an example, iPhone 7, released in September 2016, uses the *Apple A10 Fusion* 64-bit system-on-chip (SoC), which consists of two low-power cores and two high-power cores. As another example, the low-end Redmi 5 Android phone, released on December 2017 by Xiaomi, is equipped with a *Snapdragon 450* SoC carrying 8 cores. Therefore, to make full use of the computational capacity of mobile phones, especially the multi-core processors widely available on smart phones, our system must be parallelized to work on multiple cores for real-time conversion performance.

Parallel computing is a type of computation in which many calculations or the execution of process are carried out simultaneously on multiple cores. In simple words, a large complicated problem can be decomposed into smaller and easier ones, which can be allocated to different cores to be solved simultaneously, and thereby reduce the total computation time required [44]. Due to the physical constraints preventing frequency scaling, parallel computing has gained broader interest and has become the dominant paradigm in computer architecture, as shown by the popularity of multi-core processors [45]. For example, the multi-core processors have been exploited to efficiently decode videos [46]. In our voice conversion system, the computation is parallelized in mainly two spots, the speech modeling and analysis phase during both training and conversion, which is illustrated in Fig. 2.

The parallelization during the training phase is obvious and straightforward, as illustrated on the left side of Fig. 2

https://doi.org/10.1017/ATSIP.2018.23 Published online by Cambridge University Press

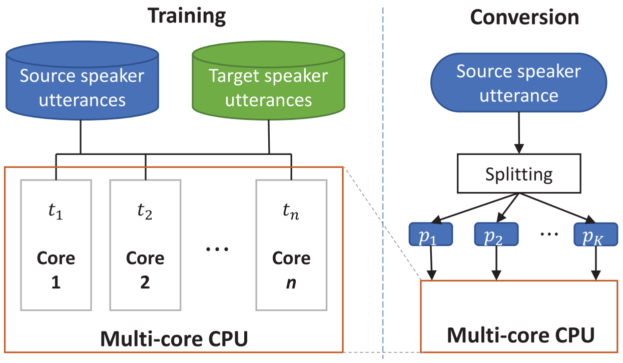
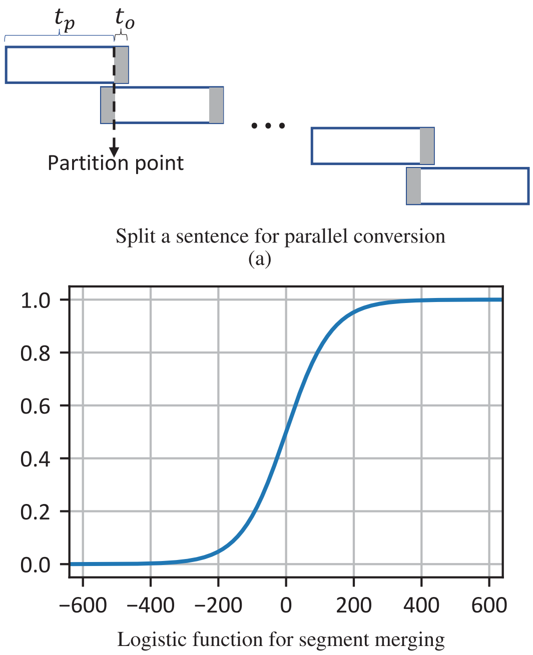


Fig. 2. Parallel computation during training and conversion phases.



Split a sentence for parallel conversion
(a)

Logistic function for segment merging
(b)

Fig. 3. Multi-core parallelism in the conversion phase. First, an input sentence is partitioned into multiple segments and those segments are converted simultaneously; then, the converted segments are merged smoothly using a logistic function for weighted sum. (a) Split a sentence for parallel conversion (b) Logistic function for segment merging.

since the speech analysis and feature construction of each utterance sample is independent. Therefore, we can distribute the total $2M$ samples from both the source speaker and target speaker on the C cores by creating a thread t_i on each core. In such a way, ideally, C audio samples can be processed at the same time without interfering each other. Therefore, the total running time is greatly reduced because the time consumed by speech analysis and feature engineering occupies a large portion of the entire time cost.

It is more subtle to parallelize the conversion phase. The average duration of sentences in the corpus is about 4 or 5 s, similar to the ones we speak everyday. Given an utterance input from the source speaker to be converted, we first split it into a group of continuous partitions p_i, $i = 1, 2, \ldots, K$, whose lengths are approximatively equal, say, around t_p seconds. Then, similarly to the training phase, we again allocate these short partitions to different cores for simultaneous processing, as shown on the right-hand side in Fig. 2. To compensate the possible artifacts introduced in the segment boundary, there is an additional overlapping region of t_o seconds' duration on each side of the segmentation point, shown in Fig. 3(a). After each partition p_i is converted into p_i^c, we merge these segments into a complete converted speech using a logistic function to achieve a smooth transition around the partition point. The logistic function used for smooth transition here is given by

$$\phi(x) = \frac{1}{1 + e^{-kx}}, \tag{16}$$

where k adjusts the steepness of the curve.

In our current implementation, the overlapping length is 40 ms, i.e., about 640 samples. Supposing we index the two overlapping regions around the partition point from -640 to 640 and choose $k = 0.015$ in the logistic function (16), shown in Fig. 3(b), the sample at index i merges the corresponding two samples, s_i^l in the left partition and s_i^r in the right partition, by

$$s_i = (1 - \phi(i))s_i^l + \phi(i)s_i^r. \tag{17}$$

Though in current implementation we mainly parallelize the speech analysis, feature construction, and frame alignment steps due to their high time consumption, it should be noted that more parallelization can still be applied to the subsequent steps such as the estimation of GMM parameters in equation (8). For example, in the work of Wojciech Kwedlo [47], a shared memory parallelization of the standard EM algorithm based on data decomposition is proposed to learn the GMM parameters on a multi-core system for higher performance. Thus, we are planning to implement such features to further accelerate our voice conversion system in its next version.

B) Vectorization via SIMD

After the computation work is spread on multiple cores, our next target is to make the single-core throughput as high as possible. Again, we need to explore and take full advantage of the hardware's capacity, especially in-core parallelism for operations on arrays (vectors) of data, due to the heavy use of matrices in the underlying algorithms. On modern mobile phones, in-core parallelism, or *vectorization*, is mainly supported by SIMD widely available on today's processors [44, 48].

Most mobile processors, including those from both iPhone and Android phones, are designed with the ARM architectures. For example, iPhone 7 uses the *Apple A10 Fusion* 64-bit system-on-chip, which is based on the ARMv8-A architecture with 64-bit instructions support. Particularly, modern ARM mobile processors, such as all of the Cortex-A8 series, usually support the Advanced SIMD extensions, known as *NEON*, which is a combined 64- and 128-bit SIMD instruction set that provides standardized acceleration for media and signal processing applications [48]. NEON can support 8-, 16-, 32-, and 64-bit integer

https://doi.org/10.1017/ATSIP.2018.23 Published online by Cambridge University Press

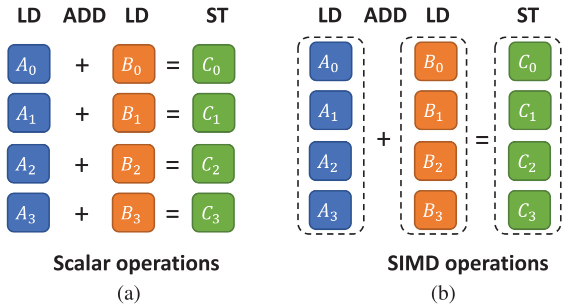


Fig. 4. Scalar vs. SIMD operation for multiple additions.

and single-precision (32-bit) floating-point data and provide SIMD operations up to 16 operations at the same time. In a study on digital image stabilization for mobile devices, the motion vector projection and the FFT calculations are accelerated using the Neon SIMD engine available on the ARM CPU, and the global motion vector for each frame can be calculated in <20 ms [49]. As its counterpart, in x86/64 architectures commonly adopted for PCs, the processors usually provide SSE or AVX-based SIMD instruction set.

In brief, SIMD performs the same operation on multiple data elements of the same type aligned in a *vector* simultaneously. Theoretically, if we can process q data elements as a whole with a single SIMD, then the speedup is close to q compared with sequential processing via scalar instructions, though the actual speedup is potentially limited by the memory bandwidth. Fig. 4 illustrates a common operation in multimedia processing, where the same value is being added to a large number of data elements, implemented with scalar instructions and SIMD instructions separately for comparison purpose. In the context of audio processing, audio data usually comes in 16-bit integer types. Supposing we are using the 64-bit NEON registers for vectorization, we can pack four 16-bit integers into this single register at the same time, as shown in Fig. 4(b). The process using only scalar instructions is depicted in Fig. 4(a). To perform a single addition, two load operations (LD) are first executed to read two numbers A and B from memory. Then, an addition (ADD) is instructed to get the result C. Finally, the sum C is written back to memory with a store (ST) instruction. By contrast, SIMD enhanced load, add and store instructions can perform on four integers simultaneously, which is highlighted in Fig. 4(b). Therefore, in total, 16 scalar instructions are reduced to only four instructions via SIMD vectorization, which yields a theoretical speedup of four.

To make use of SIMD in practical programming and software development, there are mainly three ways: (1) coding in low-level assembly instructions directly; (2) using intrinsic functions with C interfaces which internally wrap assembly codes; and (3) leveraging *auto-vectorization* if supported by compilers [48]. Generally, programming in the assembly language directly is labor-intensive and highly error-prone, despite their possibly extreme performance. On the other hand, auto-vectorization can be completely conducted by a compiler itself with no need of human intervention. However, at present even state-of-art compilers are not smart enough, and consequently, only a small portion of codes can be vectorized automatically. Therefore, our study relies on intrinsic functions as a trade-off between system performance and manual labor.

It is worth pointing out that, in a real implementation, dependence on intrinsic functions does not necessarily mean we have to call them directly in programming by ourselves. Notably, the major part of the speech analysis and conversion algorithms are formulated into matrix computations. Therefore, we can resort to some mature linear algebra libraries, which expose high-level interfaces to facilitate application development while internally making use of intrinsic functions to vectorize matrix/array computation as much as possible. That is, the internal utilization of SIMD instructions is almost transparent to users. Relevant details are described in the next subsection.

C) C++ implementation

In the above two subsections, we have described core-level and instruction-level parallelism to accelerate the core algorithms for voice conversion. The objective of this study is to develop an efficient voice conversion system on mobile devices, not limited to a specific device type like iPhone, iPad or Android phones. Though the GUI development kits differ greatly from each other on different mobile operation systems, such as iOS for iPhone and Android for most of the other smart phones, we want the core algorithm implementation of this system to be platform independent such that this crucial part can be easily ported across multiple platforms with minimum modifications. Hence, we separate the core functionality and the GUI in our system architecture. To satisfy both platform neutrality and high-efficiency demands, the C++ programming language is the best candidate for coding the core algorithms. As a general-purpose language, C++ highlights performance, efficiency, and flexibility, i.e., it allows low-level memory manipulation while still providing high-level abstractions in an object-oriented manner. Below we briefly outline how to achieve multi-core and SIMD-based parallelism in C++.

To apply thread-based parallelism for multi-core computation, that is, multi-thread programming in C++, the simplest way is to use the standard OpenMP API, which supports multi-platform shared-memory parallel programming in C and C++ [50] by only using straightforward directives. However, OpenMP is still poorly supported on iOS, the mobile operation system of iPhone or iPad. Instead, as Apple recommends, we should use Grand Central Dispatch (GCD) technology for task parallelism, which is specially designed and highly optimized for iOS. In short, those simple parallelism frameworks are still somewhat platform dependent. Fortunately, this portability problem can be solved with the new C++ 11 standard published in 2011, which adds support for multi-threading programming by introducing a new thread library. In simple words, we can create a new thread by instantiating the `std::thread` class by passing a function representing the work to be done in this thread. Special attentions should be paid to

https://doi.org/10.1017/ATSIP.2018.23 Published online by Cambridge University Press

avoiding data races in such multi-threaded programs by using synchronization primitives like `std::mutex` and `std::lock` to protect shared data. Interested readers may refer to this excellent book [51] for more details on multi-thread programming in C++.

As for vectorization on each core using SIMD instructions, we lean upon the well-known *Eigen* library, a high-performance C++ template library for linear algebra, which has been used in many industrial projects [52]. We choose *Eigen* particularly because it supports all matrix sizes and provides common algorithms for matrix decomposition, numerical equation solvers and other related algorithms such as fast Fourier transform (FFT). More importantly, *Eigen* is fast enough by performing explicit vectorization internally for SSE, AVX, and ARM NEON instruction sets while hiding this complexity from users. That is, we can achieve instruction-level parallelism as much as possible by simply implementing the algorithms using matrices and linear algebra operations provided by *Eigen*. As a side benefit, *Eigen* provides interfaces similar to MATLAB, which can simplify programming and make the codes more readable especially for engineers.

Now we can give a bird's-eye view of the software architecture of our voice conversion system shown in Fig. 5. It is clear that our system splits the core algorithms (*compute engine*) and the graphical user interface (GUI) to maximize platform portability. Ideally, no modifications of the core algorithms' source codes are required when ported to another platform, and we just need to recompile the C++ codes with the platform specific compilers for deployment. On the contrary, generally, the GUI part is tightly coupled to specific platforms, such as iOS, Android and Windows, and cannot be migrated. For example, on iOS, we construct the GUI with the *UIKit* framework using either the Swift or Objective-C language, while Android provides its own UI components accessible by the Java or Kotlin language. One pitfall of the C++ based computing kernel is that C++ only supports poor or even no interoperability with some languages including Swift, the main language for iOS development. Thus, we decide to wrap the input/output interfaces of core algorithms using the C programming language, which can interact with all mainstream languages quite easily. In the complete mobile application, the GUI communicates with the core algorithms by passing information mainly through the `train` and `convert` functions provided as C interfaces. In such a way, the computational engine, i.e., the core algorithms, can work well on all the four operation systems shown in Fig. 5 as we have tested, two for mobile devices and two for PCs, thus realizing our objective of platform independence.

IV. IOS APPLICATION DEVELOPMENT

After the core algorithms for the voice conversion system are implemented with C++, the remaining work is to develop a friendly GUI on mobile phones, i.e., to develop a mobile application, to facilitate the access of this system. As we have illustrated earlier in Fig. 5, we can develop GUI with different languages and tools on different platforms, which is well decoupled from the compute engine in our architecture. In this section, as a specific example, we present the development of an iOS application (app), which is deployed on iPhone 7, arguably the most popular smart phone in 2017. Overall, as a prototype to verify the feasibility of voice conversion on mobile phones, our app focuses on usability and concision. The most fundamental usage of this app only involves two steps depicted in Fig. 6. Firstly, both the source speaker and the target speaker read a small number of sentences while their utterances are recorded by the phone to form a parallel corpus for subsequent conversion model training. Secondly, the source speaker can speak something to the app again and it will try to convert this message to make it sound like being spoken by the target speaker. In the following, more details about the modular structure and some specific design issues of this iOS application are presented.

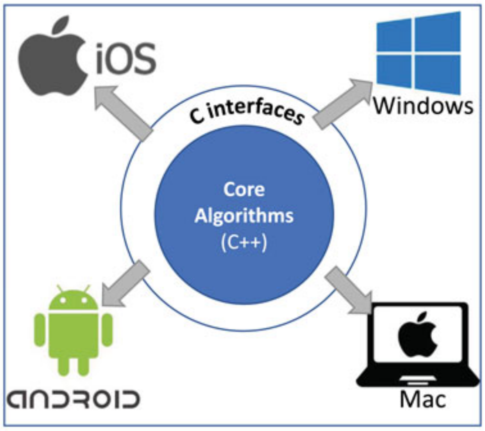


Fig. 5. Overview of software architecture. The core algorithms (compute engine) are implemented in C++ for portability. The graphical user interface (GUI) sits on top of the engine and may be built with different languages/libraries on various platforms. The engine exposes C interfaces to be used by GUI on four common operation systems: iOS, Android, Windows, and Mac.

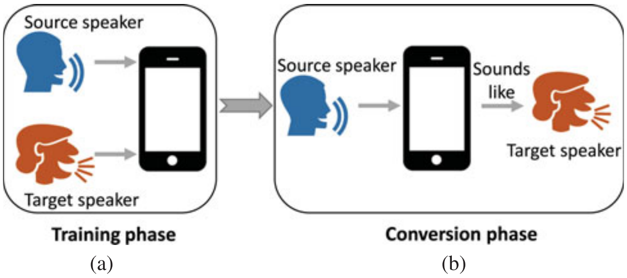


Fig. 6. Workflow of the voice conversion application on mobile phones.

A) Overview of the functional modules

In software engineering, it is well known that modularization is a fundamental principle for large-scale software

https://doi.org/10.1017/ATSIP.2018.23 Published online by Cambridge University Press

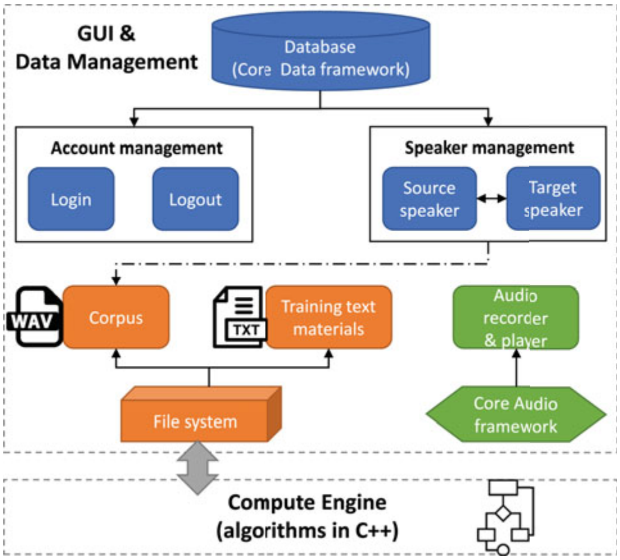


Fig. 7. Outline of the modules in the iOS application.

development. The main modules in our system are exhibited in Fig. 7. To manage the user accounts, including registration, login and logout, we store the user information in a light-weight mobile database such as SQLite, whose access is mediated by the iOS Core Data framework to simplify coding. Similarly, for one source speaker (i.e., the user), multiple target speakers can be trained, and the association between the source speaker and the target speaker are also recorded into the database. Most of the hard disk storage of this app is occupied by the corpus built for voice conversion training. We provide text materials each containing about 20 normal sentences to be read by a pair of source speaker and target speaker. While a speaker is reading the transcript, his/her voice can be easily recorded by our app (Fig. 6) and saved as monophonic WAV files with a sampling frequency of 16 kHz, with support from the iOS Core Audio framework.

At last, it should be noticed from Fig. 7 that currently in our system the interaction between the GUI layer and the compute engine, i.e., the core algorithms implemented in C++, is quite clear and concise. During training, the GUI layer only needs to tell the engine what are the WAV files composing the training set, and the engine will produce a model file corresponding to the conversion function. Similarly, at the point of conversion, the GUI layer notifies the engine about the audio file from the source speaker to be converted, and the engine will finally store the finished converted speech in a new WAV file and inform the GUI layer of the file path. We can hear the converted voice by simply playing this audio file. Besides, it can also be set to be played automatically when finished.

B) Specific design issues

It is a very challenging task to develop a well-performing iOS application, which involves UI design, business logic modeling, coding with Swift/Objective-C, debugging with XCode, and many other tedious works. Here, details are omitted to conserve space. In the following, we only list and address two specific design concerns about the app development as representatives of the whole process.

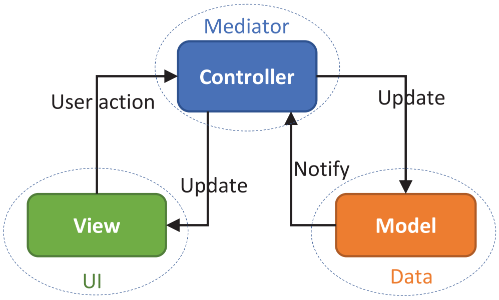


Fig. 8. Model-View-Control(MVC) design pattern.

1) MVC DESIGN PATTERN FOR THE USER INTERFACE

In developing GUI-based applications, a well-known best practice is to separate the data from its presentation, because the presentation may vary a lot according to specific requirements, e.g., in a chart or in a table. Since the first release of iOS, Apple has recommended to adopt the Model-View-Controller (MVC) design pattern to emphasize this best practice, which is delineated in Fig. 8. Generally, model objects hold the data or knowledge we have, and view represents something visible in the user interface dedicated to displaying the data. However, the model and the view cannot communicate directly in order to reduce coupling. Instead, they are mediated by the controller object, typically through the delegate pattern.

In practice, we define a model object to access the data stored in the database or the file system, as shown in Fig. 7. When there is a proper change in the data, the model notifies the controller object, and subsequently, the controller attempts to update the view according to the changed data. In the other direction, when the user interacts with the view, his/her action is passed to the controller, which further informs the model about the user intention, e.g., updating or deleting data. Despite its simplicity, the layers of a typical GUI application can be cleanly decoupled by the MVC pattern to encourage better organization and to promote code reuse. In our app, each window is constructed with the MVC pattern described above.

2) DATA-DRIVEN PRESENTATION IN COLLECTION VIEWS

To better display multiple audio information and target speaker items, table view and collection views are extensively used in the GUI interface of our application. For better flexibility and reusability of the program, it is best to separate the data from its visualization through a UI-Data-Operation style. In the iOS development framework, the data are stored in a data source object and the operation is represented by a delegate object. Note that the table views and collection views in iOS *UIKit* are all designed with a specialized MVC mechanism.

https://doi.org/10.1017/ATSIP.2018.23 Published online by Cambridge University Press

The data source is only responsible for providing the data and does not know how the data are displayed. The delegation gives objects a chance to coordinate their appearance and state with changes occurring elsewhere in a program, usually brought about by user actions. More importantly, delegation makes it possible for one object to alter the behavior of another object without the need to inherit from it. With this design pattern, we can better decouple the data source and its presentation. Once the data are updated somewhere, the UI will respond automatically to reflect the data changes. Hence, such data-driven presentation style is also widely adopted in our app to facilitate the complex user interface development.

V. EXPERIMENTS AND RESULTS

We have developed the voice conversion system in a convenient bottom-up manner since the two layers in Fig. 7 are almost perfectly decoupled. That is, the compute engine responsible for intensive computation was first constructed with a focus on accuracy and efficiency by iterative testing and improvement. The next step was designing the user interface and the associated data managers for a mobile application with attention paid to usability and conciseness. The finished mobile application, called *Voichap*, was successfully deployed on an iPhone 7 device. In the following, we first evaluate the core algorithm speedup enabled by multi-core and vectorization-based parallelism. Then, we describe the user interface of this application and demonstrate its usage through specific examples. Finally, we evaluate the efficiency and effectiveness of this mobile voice conversion system by measuring its running time and conducting conversion quality test scored by 10 listeners.

A) Core algorithm speedup test results

To evaluate the efficiency of our implementation making full use of parallelism, we tested the engine performance with various environment configurations. For simplicity, we conducted the tests on a Windows 10 PC with an Intel Core i7 CPU carrying 4 hardware cores, since the algorithms were first developed and tested with the powerful Visual Studio IDE on Windows 10. This also demonstrates that our core algorithm implementation is indeed platform agnostic, since we can run the same code on iOS (iPhone 7) with no changes at all. In each test measuring the running time listed below, the experiments were repeated for 10 times for more accurate estimation.

1) Overall performance

We timed the training process for 20 utterances, each with a duration of 4.5 s on average, from both the source speaker and target speaker, which is the default configuration of our mobile application. In this test, the number of Gaussian components used in the model (7) is chosen as $m = 4$. As the baseline, we also measured the running time of the original MATLAB code on the same PC. It should be reminded here that many built-in functions of MATLAB are in fact highly optimized C or Fortran routines, which may be well implemented in a parallel manner internally. Thus, it is common that a naive C++ numerical program cannot beat its counterpart in MATLAB in time efficiency. The statistics of the measured running time is illustrated in Fig. 9. Additionally, we should note that the frequency warping method only transforms the voiced frames. Thus, the number of voiced frames in the utterances may be of more interest. In our training set including 20 utterances, the average number of voiced frames in each utterance is 302.6 and the frame shift is 8 ms (128 samples). Thus, the average length of the voiced signal in the training set is only 2.42 s. Similarly, we can count the average number of voiced frames in the test set for conversion, which is about 297.5, corresponding to an average duration of 2.38 s if we only consider the voiced segments.

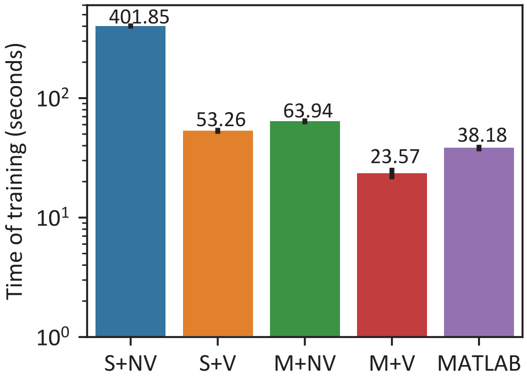


Fig. 9. Running time of the training phase with different configurations (95% confidence interval). From left to right, S+NV: single-threaded C++ with no vectorization, S+V: single-threaded C++ with vectorization, M+NV: multi--threaded C++ without vectorization, M+V: multi-threaded C++ with vectorization, MATLAB: 64-bit MATLAB 2016 with default settings.

As is shown in Fig. 9, the C++ program enhanced by both vectorization and multi-core parallelism can even outperform the one in MATLAB, which is already quite fast due to its highly optimized built-in functions and specific toolboxes devoted to signal processing. This result also demonstrates the inherent efficiency of C++ as well as the high performance of *Eigen*. Obviously, when we make the most of the hardware characteristics to parallelize the program, including both multi-core and vectorization-based parallelism, the running time is reduced to a minimum around 25 s. Of course, the precise running time on mobile phones will be longer than the one on PC due to the limited power of mobile devices. However, the acceleration effects of hardware-enabled parallelism on system performance are similar, indicating the necessity of parallelizing the core algorithms for truly efficient implementation on mobile phones.

2) Profiling of each stage

In the current implementation of the training phase, the first two stages including speech analysis, feature construction, and frame alignment (see Fig. 1) are parallelized over multiple cores. Ideally, each utterance or pair of source-target

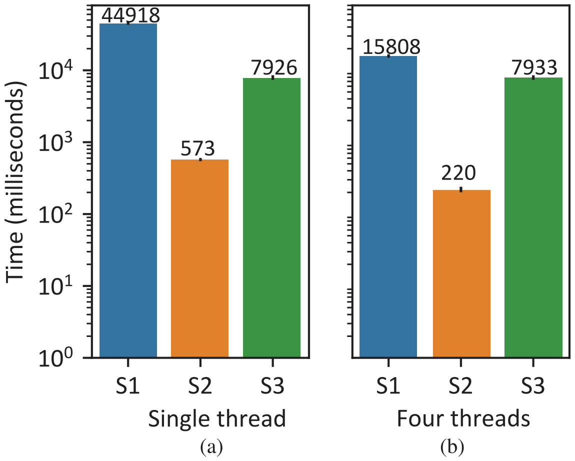


Fig. 10. Running time of each stage during the training phase with different configurations (95% confidence interval). S_1: speech analysis and feature construction; S_2: frame alignment of parallel corpus; S_3: conversion function training. Vectorization is enabled for all conditions.

utterances (for frame alignment) can be processed independently and simultaneously on separate cores. To further inspect the workload distribution, we measured the running time of each stage during training with 20 parallel training samples, each with a duration of about 4.5 s on average. The results are plotted in Fig. 10(a).

Clearly, it is shown that the first stage, i.e., speech analysis and feature construction, takes most of the time (about 84%). By contrast, the second stage of training, i.e., frame alignment only takes a negligible proportion of the total time for the current training set. As for the final stage, conversion function training, though it takes a non-negligible amount of computation time, it is not straightforward to parallelize this stage, because it requires all the data as a whole. Furthermore, since the last stage takes much less time compared with the first two stages, we choose to only implement the first two stages through multi-core parallelism for now. By distributing the workload of the first two stages to four cores using four threads, the running time of these two stages is greatly reduced to about 35.2% and the total running time is reduced to about 44.8% of the time needed on a single core, which is demonstrated in Fig. 10(b). Finally, we want to emphasize that, though the time spent by the frame alignment stage (S2 in Fig. 10) seems to be minor, it is still necessary to parallelize this stage, because its theoretical time complexity is quadratic due to the use of dynamic time warping [53], while the other two stages only demonstrate linear time complexity empirically.

3) Effect of number of cores

To further analyze the speedup effect brought by multi-core parallelism, we measured the training time for the training set composed of 20 parallel utterances of average duration about 4.5 s as well as the conversion time of a 5-s source input utterance. Since there are in total 4 cores in the CPU of our PC, the program was configured to use a different number of cores ranging from 1 to 4 for parallelism. The running time with respect to the number of cores allowed is reported in Fig. 11. Since only the first two stages of the training phase benefit from multi-core parallelism, that is, about 84% of the program can be parallelized (Fig. 10(a)), the acceleration effect exhibited in Fig. 11 is indeed consistent with the Amdahl's law, which is often used in parallel computing to predict the theoretical speedup when using multiple processors [54]. Specifically, with four cores available, the speedup of the training phase is about 2.3 times, while Amdahl's law predicts a theoretical speedup of at most 2.7 times. Overall, according to the relationship between running time and the number of cores shown in Fig. 11, this voice conversion system is characterized by sublinear speedup when multi-core parallelism is applied.

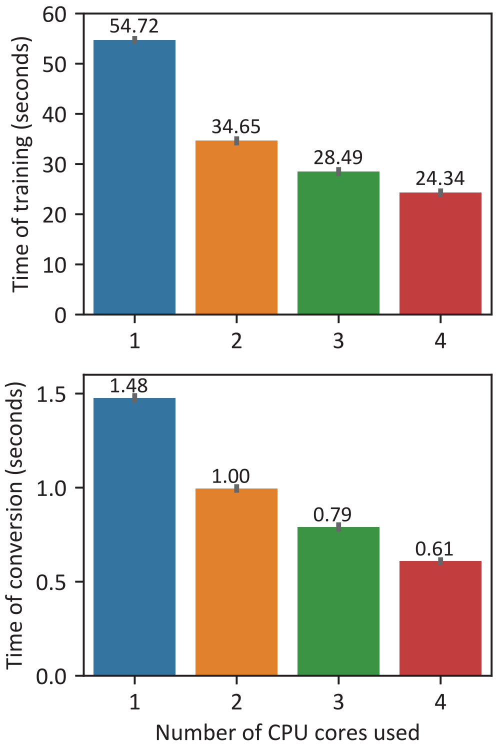


Fig. 11. Running time of the training phase (top) and the conversion phase (bottom) when a different number of CPU cores are used for multi-core parallelism (95% confidence interval). There are four cores in total in the CPU under investigation. Vectorization is enabled.

We further investigated the scalability of multi-core parallelism in our system by increasing the training set size proportionally to the number of cores used. Results are shown in Fig. 12. Obviously, the average time required for 10 training samples per core increases as the total problem size enlarges. This is due to the fact that not all the parts of the algorithm framework are parallelized (Fig. 10). Nevertheless, the utilization of multi-core parallelism in our system is still scalable, since the running time can be greatly reduced for a fixed problem size when more cores are available (Fig. 11). From a practical perspective, it should be noted that though in principle we can use as many threads as we like when implementing multithreading based parallelism, the speedup cannot be further enhanced once there are more threads than cores on a single CPU. For example, in

https://doi.org/10.1017/ATSIP.2018.23 Published online by Cambridge University Press

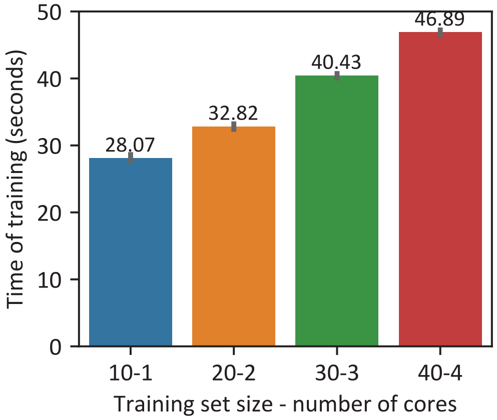


Fig. 12. Training time of various training set sizes and numbers of cores (95% confidence interval). On average, each core corresponds to a training set of 10 utterances. A label *n*-*k* means *n* utterances in the training set and *k* cores. Vectorization is enabled.

our case the training phase of our system is CPU-intensive, and thus at most four threads can be executed simultaneously on a 4-core CPU.

B) Demonstration of the application

The mobile application we have developed for iPhone 7, *Voichap*, has six windows in total, whose screen shots are shown in Fig. 13, which correspond closely to the various modules in Fig. 7. The most important actions are performed in Figs 13(d) and 13(f), where the training and conversion take place. We have provided some transcripts to be read by both the source speaker and the target speaker to build a parallel corpus for training purpose, as shown in Fig. 13(d), where we can also edit the target speaker information, including his/her name and a short description. More specifically, in the context of *Voichap*, the source speaker actually refers to the user. Hence, the user only needs to read these sentences once and the saved audios can be reused to match all target speakers for training in the future. Afterward, the user can ask the target speaker, e.g., his/her friend, to read these sentences again to the phone and the utterances are recorded by this app to create the training corpus.

As previously illustrated in Fig. 7, the speaker information is stored in a light-weight database and the taped audios are organized in a specific directory for each speaker in the iOS local file system. After we finish the recording, the *Train* button in Fig. 13(d) can be touched to start the training process, and at the end, a conversion model file is generated corresponding to the given source-target pair. Note that the model file contains all the necessary information required to quickly restore the conversion function $f(\cdot)$ when needed (Fig. 1). Now, it is time to play with real-time voice conversion highlighted by the *Speak here* screen in Fig. 13(f). Just simply press and hold down the microphone button, and say anything you like. Once the microphone button is released, the conversion phase starts automatically and immediately. After a short while, you can hear what you have just spoken being repeated, but in the voice of the target speaker.

As a side note, our *Voichap* application can also be fed with audio files directly aside from recording improvizational utterances by ourselves. This support can make the application much more interesting. As aforementioned in the introduction section, currently, there are some paid apps available on Google Play or Apple Store regarding voice changing, which can mimic a celebrity's voice though generally the performance is poor. However, with our *Voichap* app, you can transform your voice into the one of any celebrity once you can get some training utterances from him/her. Of course, it is impractical to really ask a celebrity to read the specified text to our phone. However, we can easily download their public speeches from websites like YouTube and then extract the audios and break them into sentences as the training materials of the target speaker. The remaining work is to speak the same sentences and record our utterances as the source speaker's training data. This process is shown in Fig. 14, where we take President Trump as an example. In short, a new target is first created and the audio files are loaded, which we grabbed from YouTube. After the model is trained, we can speak words like 'make America great again' as if we were President Trump while the voice conversion system modifies the voice individuality and makes it as if it was really spoken by President Trump.

C) Experimental evaluations of the voice conversion system

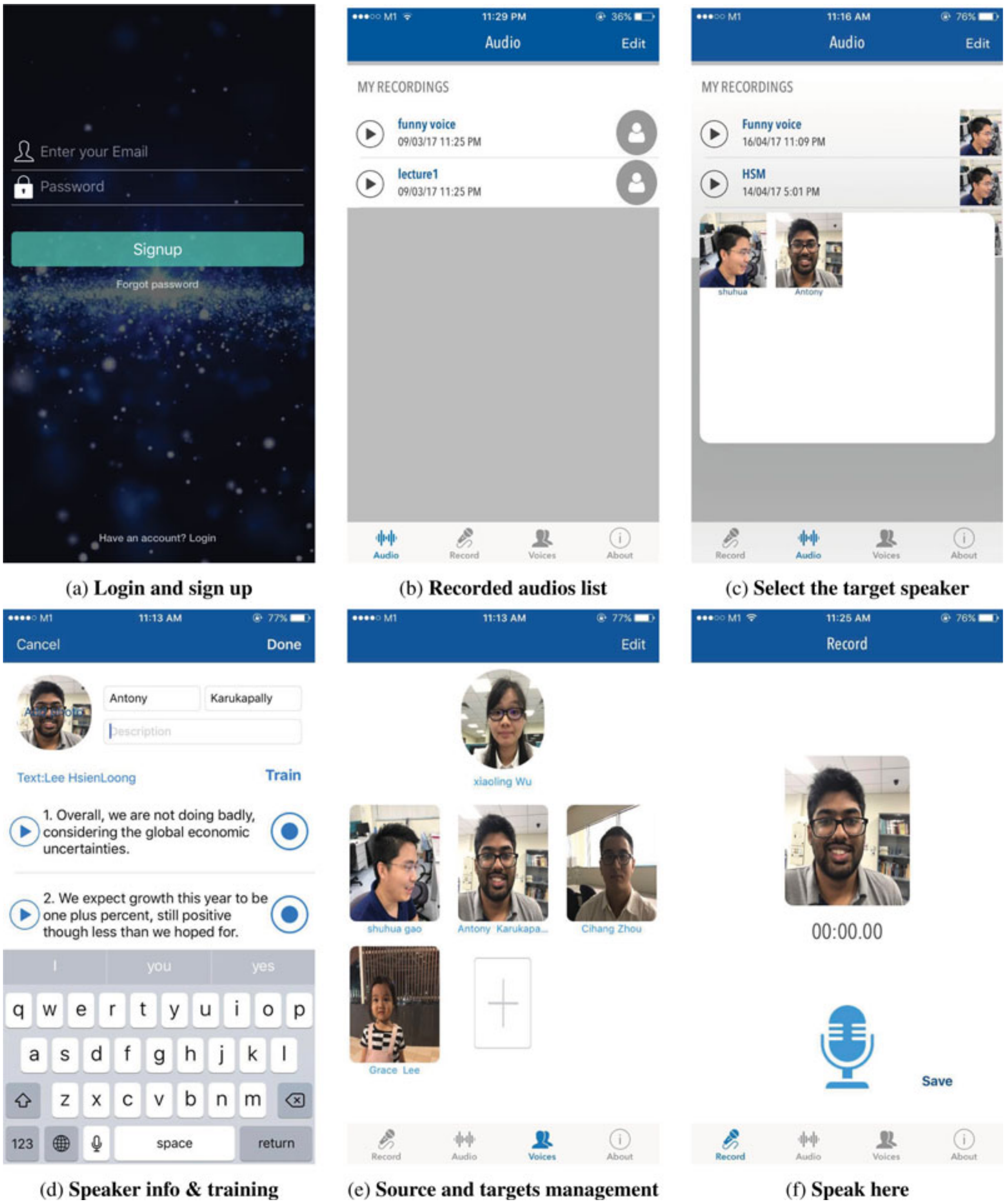
The effectiveness and real-time performance of our mobile voice conversion system, *Voichap*, were evaluated with voice conversion among both male and female speakers. Unlike the traditional benchmarking of voice conversion, which is always pursuing better transformation similarity or speech quality regardless of the computational cost, as a practical mobile app, *Voichap* attempts to achieve good but not necessarily the best performance yet in an acceptable period of time. Our ultimate objective is to develop a real-time voice conversion system on mobile phones for daily use and the user experience matters most. Obviously, it does not make much sense in this situation if a user has to wait for half an hour to get the converted speech even if the conversion quality is extremely high.

Four candidates ranging in age from 20 to 30 were chosen as speakers in the following tests, including one male and one female as the source and the same setting for the target. Thus, four speaker pairs are examined in total: male-to-male (M–M), male-to-female (M–F), female-to-male (F–M), and female-to-female (F–F).

1) Parameter tuning and time efficiency

The two major parameters of our voice conversion system are the number of training sentences and the number of Gaussian components in the GMM model, i.e., *m* in equation (7). According to a survey of multiple users, we found that it is a good choice to build a transcription collection of 20 sentences with a duration around 4 or 5 s for

https://doi.org/10.1017/ATSIP.2018.23 Published online by Cambridge University Press



1. Overall, we are not doing badly, considering the global economic uncertainties.

2. We expect growth this year to be one plus percent, still positive though less than we hoped for.

(a) Login and sign up (b) Recorded audios list (c) Select the target speaker

(d) Speaker info & training (e) Source and targets management (f) Speak here

Fig. 13. User interface of the *Voichap* application on iPhone 7. (a) Login and sign up (b) Recorded audios list (c) Select the target speaker (d) Speaker info & training (e) Source and targets management (f) Speak here.

the purpose of training. Though generally a larger training set can contribute to better performance, it may make the user get bored to read a lot of sentences and can also lead to an overlong training time. To determine the best option of m, we mainly need to balance the running time and the conversion performance. By testing various values of m, we summarize the training time with respect to each m in Fig. 15.

Clearly, the training phase will take a longer time if more Gaussian components are adopted. Besides, we see that the total training time exhibits a nonlinear relation with respect to the number of Gaussian components m. On the other hand, the average training time for the training set including 20 utterances ranges from 20 to 55 s for various m's. Most importantly, through a subjective listening test of the converted speech in terms of converted-to-target similarity and speech quality, which is illustrated in Fig. 16, we find that the overall improvement is minor once $m > 4$ and consequently the listeners give comparable opinions on the similarity and quality of the converted speeches generated by various m values above this threshold. Therefore, from Figs 15 and 16, it is sensible to choose $m = 4$ as the default parameter value of our system, which can help greatly reduce the running time yet without significantly deteriorating the converted speech quality and similarity from the aspect of human perception. In the following objective and subjective evaluations, we all set $m = 4$.

https://doi.org/10.1017/ATSIP.2018.23 Published online by Cambridge University Press

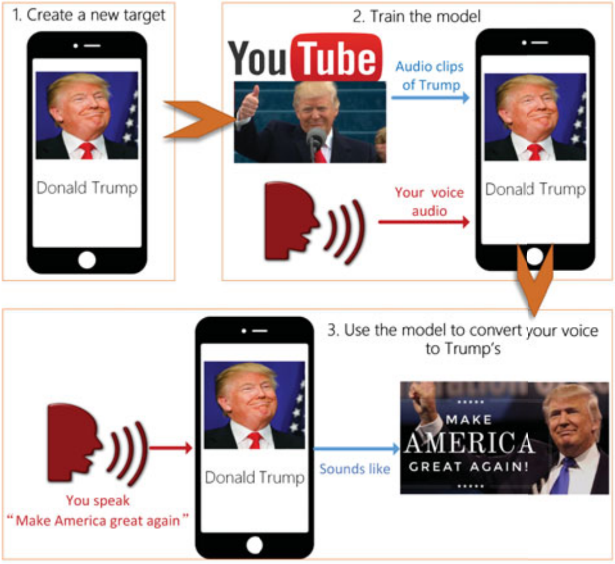


Fig. 14. A typical usage scenario: how to make yourself sound like President Trump?.

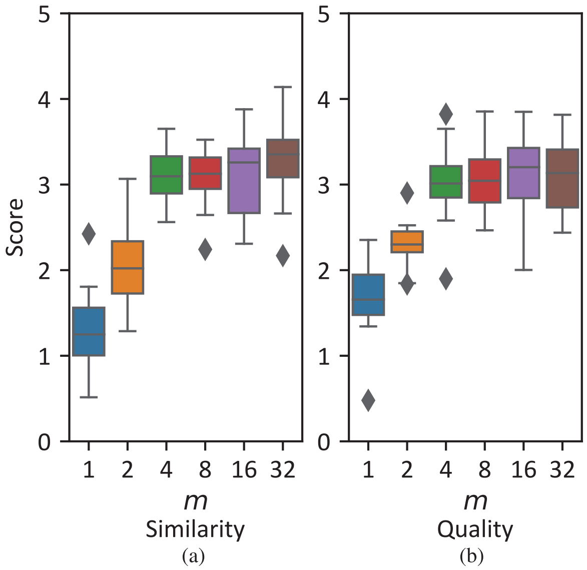


Fig. 15. Measured running time of the training phase with respect to different number of Gaussian components (95% confidence interval). The training set contains 20 sentences of average length around 4.5 s and the running time is measured by repeating 10 times. Vectorization and 4-core parallelism are enabled.

Let's return back to the mobile application *Voichap* now. Using the same training set and test set, the running time of the app with both multi-core and multi-thread parallelism enabled on iPhone 7 was measured as follows:

- The training phase of a parallel corpus containing 20 sentences of average length about 4 s takes approximately 36.5 s.
- The conversion phase for a 4.6-s sentence from the source speaker takes about 0.85 s.

This result meets our expectations on the basis of Figs 9 and 15. Note that compared with a PC, the computational ability of an iPhone device is relatively poor, which inevitably leads to a longer running time than the one measured with MATLAB on a Windows 10 PC.

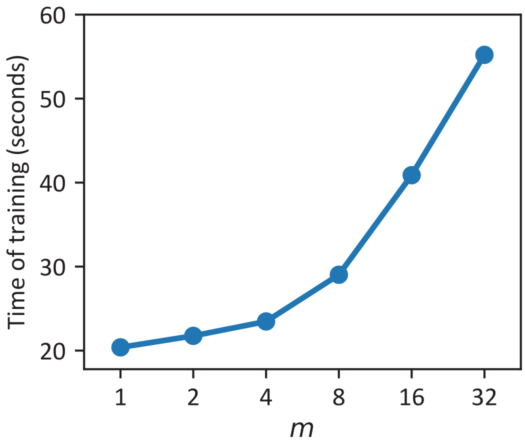


Fig. 16. Subjective evaluation of the voice conversion system with respect to the number of Gaussian components m in GMM. 10 listeners were asked to evaluate the speech similarity and quality. Here it shows the average score of the four possible source-target conversion directions.

Table 1. The MCD of the unconverted source, the traditional GMM and the weighted frequency warping (WFW) method

	No conversion	GMM	WFW
MCD(dB)	7.83	5.85	5.97

2) Objective evaluation of voice conversion

Usually, two kinds of evaluations can be used to score the performance of a voice conversion system: objective evaluation and subjective evaluation. For objective evaluation, the most widely used measure in literature is the Mel-cepstral distortion (MCD) between the converted speech and the original target speech [9, 15, 18]. The MCD is calculated by

$$\mathrm{MCD[dB]} = \frac{10}{\log 10}\sqrt{2\sum_{i=1}^{24}(c_i - \hat{c}_i)^2}, \tag{18}$$

where c_i and $\hat{c}_i$ are the ith coefficients of the target and converted Mel-cepstral coefficients (MCCs). For each pair of target-converted utterances, the MCD is calculated frame-by-frame after alignment and then averaged over all paired frames. For comparison, we also reported the MCD for the traditional GMM method [17]. The results are listed in Table 1.

Generally, a lower MCD indicates smaller spectral distortion, i.e., higher conversion accuracy. Table 1 shows that the MCD of WFW is only slightly larger than the one of GMM. This result illustrates one of the design goals of WFW, that is, it aims to improve the quality of the converted speech without decreasing the conversion similarity. In particular, this merit is largely attributed the GMM-based energy correction filter given in (14), which can slightly correct the gross details of the spectral shape such as the spectral tilt and energy distribution but without altering the small details in the wrapped spectrum [26].

https://doi.org/10.1017/ATSIP.2018.23 Published online by Cambridge University Press

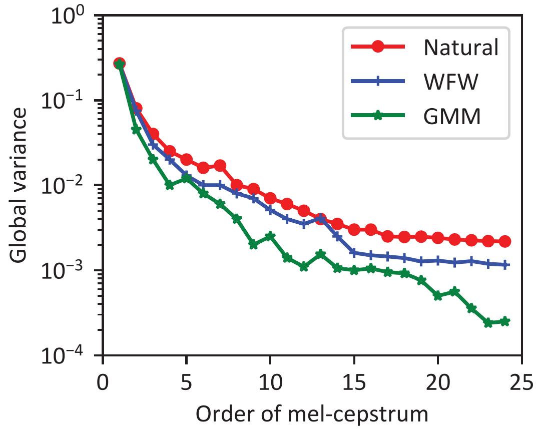


Fig. 17. Global variances for the natural speech and the converted speeches via GMM and WFW.

As aforementioned, the WFW used in our system is a combination of GMM and frequency warping. Compared with the statistical transformations like GMM, frequency warping can achieve good scores on conversion quality by overcoming the over-smoothing effect in traditional GMM. To show that WFW can contribute to more natural voice conversion, though in the last step GMM transformations are applied to correct the energy of the warped spectra, we evaluated the global variance (GV) of the converted speech, defined as follows [18],

$$\boldsymbol{v}(\boldsymbol{y}) = [v(1), v(2), \ldots, v(D)]^T,$$
$$v(d) = \frac{1}{T}\sum_{t=1}^{T}\left(y_t(d) - \frac{1}{T}\sum_{t=1}^{T} y_t(d)\right)^2, \qquad (19)$$

where $y_t(d)$ is the dth component of the target static feature vector at frame t. Similar to the MCD index (18), the Mel-ceptral coefficients are used here. The GV is calculated for each utterance and the average values are reported in Fig. 17 to compare the WFW and GMM methods.

Obviously, we see that the GV obtained by the conventional GMM approach is significantly smaller than the natural speech, indicating that the converted Mel-cepstra are excessively smoothed. On the other hand, the GV produced by the WFW method is clearly larger than the one of GMM and is closer to the GV of natural speech. This comparison shows that WFW can effectively alleviate the over-smoothing problem present in GMM by introducing frequency warping and thus the output speech is less muffled. Overall, the GV results imply that WFW can lead to more natural speech conversion than GMM, which is further verified in the subjective evaluation below.

3) Subjective evaluation of voice conversion

In the objective evaluation, we have measured and compared the MCD and the GV indices. However, it is known that the MCD metric may correlate poorly with human perception in reality, i.e., a small MCD value does not necessarily indicate a better result when evaluated by human listeners. This is known as the *perceptual deficiency* problem, which states the reduction in the mean square error between the converted and the original speech parameters may not lead to better-perceived speech [55]. Thus, currently, the objective criterion like MCD is only suitable for defining the training objective and validation purpose, but not for evaluating the final system [9]. Thus, to evaluate the true performance of *Voichap* on iPhone 7, we invited 10 volunteers to participate in a subjective evaluation test focusing on the following two aspects.

- *Speech quality*. This quantifies the quality of the converted speech in terms of naturalness, intelligibility, and presence of audible artifacts with a 5-point scale from 1 (very bad) to 5 (excellent).
- *Speech similarity*. Subjects are given a pair of speeches in random order including the original one and the converted one, and they need to score the similarity using a 5-point scale, from 1 (very different) to 5 (identical).

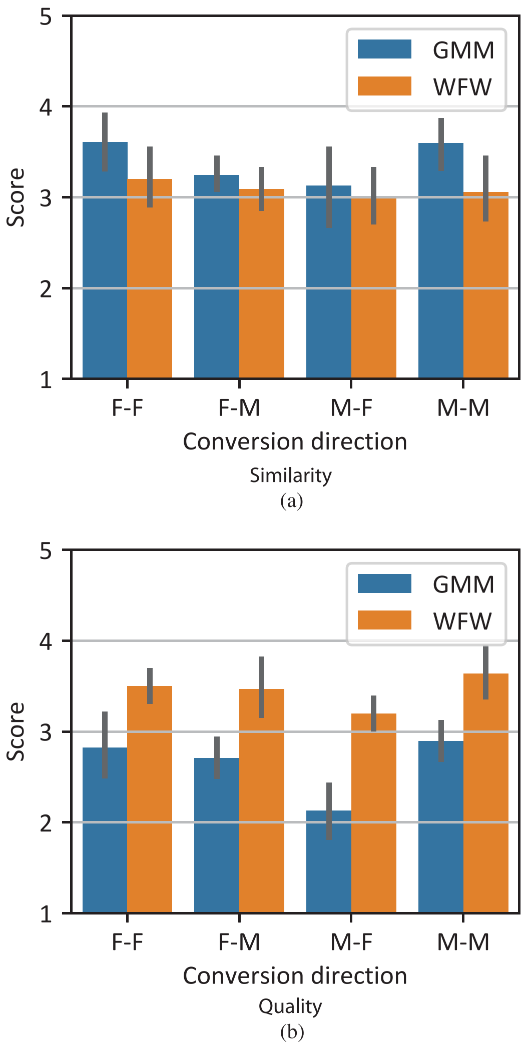
For this subjective evaluation, the training set contains 20 utterances of average length about 4 s and the number of Gaussian components is set to $m = 4$. In total, we performed four tests including two intra-gender conversions (F–F, M–M) and two cross-gender ones (F–M, M–F). The evaluation results from 10 listeners with a 95% confidence interval are shown in Fig. 18. It is not supervising that intra-gender conversions usually give better performance than cross-gender ones due to the greater difference between female and male voices. Besides, we also see that the perceptual score depends largely on individual listeners [9], as indicated by the large variance in Fig. 18. Comparing GMM and WFW, we see that there is only a small gap between the similarity scores of the two methods. However, there is a significant increase in quality from GMM to WFW, which shows the effectiveness of frequency warping in improving conversion quality. This quality enhancement is consistent with the higher GV in Fig. 17. It should be noted that in WFW by tuning the smoothing-in-frequency window in the energy correction filter (14), different balance between quality and similarity can be achieved to meet the practical performance expectation.

In general, an average score of 3 represents an acceptable level of speech quality and similarity for voice conversion. In view of the small training set in these tests, it is reasonable to conclude that we have verified the feasibility of real-time voice conversion on mobile phones. Besides, thanks to the time efficiency of this app, the user experience is also positive. Particularly, the conversion phase for a sentence of a normal length only takes <1 s, which can be considered close to real-time conversion on mobile phones.

VI. DISCUSSION

Nowadays, artificial neural networks, especially deep learning, are quite popular and it seems they can solve questions in many fields including voice conversion [24, 25, 55]. Generally speaking, deep learning-based approaches can lead

https://doi.org/10.1017/ATSIP.2018.23 Published online by Cambridge University Press



Similarity
(a)

Quality
(b)

Fig. 18. Subjective evaluation of the voice conversion system *Voichap* on iPhone 7 in terms of converted speech quality and similarity (95% confidence interval). There are 10 volunteer listeners participating in these tests and the 5-point scale is used for scoring. (a) Similarity (b) Quality.

to better results than traditional methods like the one used in our study. Thus, one may be wondering why we do not adopt deep learning for voice conversion here. It should be noted that deep learning is not a panacea because its great performance depends significantly on a huge size of training data, which unavoidably results in an extremely high computational cost. That is why a powerful GPU is often required to train a deep neural network with hundreds or even thousands of parameters. In fact, one of our authors, Dongyan Huang, has also conducted a study on statistical parametric speech synthesis using generative adversarial networks (GAN), which is closely related with voice conversion, and as many as 8000 sentences are used for network training [55]. Obviously, it is not practical to obtain such a large dataset from a normal user, not to mention the seemingly endless training on mobile devices.

By contrast, the objective of our study is to develop a *real-time* voice conversion system on mobile devices with no need for server-side support. For a mobile application, it would be insane to ask the user for such a huge amount of training audios. What is worse, the poor GPU capacity on most mobile phones can hardly accelerate the neural network training process. As a result, the network training on a mobile phone even with a moderate size of data may take hours or even days, which is definitely unacceptable, let alone the longer conversion time. Therefore, for this project, we try to implement a traditional voice conversion algorithm, the GMM-based weighted frequency warping method [33]. Even so, great efforts must be taken to implement such algorithms wisely to ensure its time efficiency such that the training and conversion can be finished in a reasonable time on mobile phones. Besides, as we have described before, a trade-off between the best possible performance and the time efficiency must be made.

One limitation of the current system is that it only supports one-to-one voice conversion and parallel corpora are required. That is why we have supplied some fixed text materials to be recorded in our application, *Voichap*. If a new target speaker is to be added, then his/her voice has to be recorded by reading these text materials to build a parallel corpus with the source speaker (the user). Such a requirement of parallel corpora may not be user-friendly enough. To make the framework more flexible and to relieve the need of parallel corpora, more advanced techniques like one-to-many or many-to-one voice conversion methods can be adopted, for example, eigenvoice conversion [56] and tensor representation [57]. Such studies deserve to be exploited in the future development of our mobile voice conversion system.

VII. CONCLUSION

Voice conversion can be applied in many practical problems, e.g., personalizing TTS systems, speech-to-speech translation, and speaking- and hearing-aid devices. Due to the prevalence of smart phones in recent years, in this paper, we tried to verify the feasibility of real-time voice conversion on a single mobile device, i.e., both the training and conversion are executed on the mobile device with no remote support from a server. Specifically, we presented the efficient implementation of the core algorithms and the development of a mobile application, *Voichap*, on iPhone 7. The most important lesson learned is to make full use of the hardware advantages commonly available on mobile phones, especially multi-core parallel computing and vectorization using SIMD instructions. The experimental evaluations show that our system can work properly on an iPhone 7 device. The training phase for a training set composed of 20 sentences takes about 40 s, while the conversion phase of a 5 s source input needs slightly more than 1 s. Besides, according to the subjective evaluations, the performance of *Voichap* in terms of converted speech quality and similarity on iPhone 7 is also satisfactory. Though the voice conversion performance of this system is definitely not the best, especially compared with state-of-the-art deep learning-based methods, it can work totally off-line and achieve real-time voice conversion

https://doi.org/10.1017/ATSIP.2018.23 Published online by Cambridge University Press

on mobile phones due to the optimized implementation, which, as far as we know, is pioneering in this field.

Future work includes adding more functionality to the current iOS app and porting it to Android phones. This should not be a difficult task thanks to the sensible design of the software architecture shown in Fig. 5, where the compute engine is well decoupled from the GUI layer. Given the rapid growth in computational capability of mobile phones, it is promising that our voice conversion app can work even faster on newly released smart phones and, ideally, deep learning based approaches will become practical on mobile devices in the near future. Finally, it should be noted that the weighted frequency warping algorithm used in our system belongs to the most commonly used trajectory-based conversion approaches, which convert all spectral parameters of a complete utterance simultaneously. To further reduce the delay of voice conversion, frame-based approaches capable of converting spectral parameters frame by frame are more desirable, e.g., the time-recursive conversion algorithm based on maximum likelihood estimation of spectral parameter trajectory [58], which is also planned in our future work.

ACKNOWLEDGMENT

This study was conducted via collaboration between Department of Electrical & Computer Engineering, National University of Singapore and Institute for Infocomm Research, A*STAR, Singapore. We acknowledge the support from both sides. We thank Mr. Mingyang Zhang from Southeast University, China, for his assistance and helpful comments on objective evaluations of the voice conversion system.

FINANCIAL SUPPORT

This research received no specific grant from any funding agency, commercial or not-for-profit sectors.

STATEMENT OF INTEREST

None.

REFERENCES

[1] Daniels, J.; Ha, L.K.; Ochotta, T.; Silva, C.T.: Robust smooth feature extraction from point clouds. *Proceedings - IEEE International Conference on Shape Modeling and Applications 2007, SMI'07*, 2007, 123–133.

[2] Kain, A.B.; Hosom, J.-P.; Niu, X.; van Santen, J.P.; Fried-Oken, M.; Staehely, J.: Improving the intelligibility of dysarthric speech. *Speech Commun.*, **49** (9) (2007), 743–759.

[3] Kobayashi, K.; Toda, T.; Neubig, G.; Sakti, S.; Nakamura, S.: Statistical singing voice conversion with direct waveform modification based on the spectrum differential, in *Fifteenth Annual Conf. of the Int. Speech Communication Association*, 2014.

[4] Kawanami, H.; Iwami, Y.; Toda, T.; Saruwatari, H.; Shikano, K.: Gmm-based voice conversion applied to emotional speech synthesis, in *Eighth European Conf. on Speech Communication and Technology*, 2003.

[5] Akanksh, B.; Vekkot, S.; Tripathi, S.: Interconversion of emotions in speech using td-psola, in Thampi, S.M.; Bandyopadhyay, S.; Krishnan, S.; Li, K.-C.; Mosin, S.; Ma, M.: Eds., Advances in Signal Processing and Intelligent Recognition Systems, Springer, Trivandrum, India, 2016, 367–378.

[6] Felps, D.; Bortfeld, H.; Gutierrez-Osuna, R.: Foreign accent conversion in computer assisted pronunciation training. *Speech Commun.*, **51** (10) (2009), 920–932.

[7] Aryal, S.; R., Gutierrez-Osuna: Can voice conversion be used to reduce non-native accents? in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE Int. Conf. on. IEEE*, 2014, 7879–7883.

[8] Wu, Z.; Li, H.: Voice conversion versus speaker verification: an overview. *APSIPA Trans. Signal. Inf. Process.*, **3** (2014), e17.

[9] Mohammadi, S.H.; Kain, A.: An overview of voice conversion systems. *Speech Commun.*, **88** (2017), 65–82.

[10] Kawahara, H.; Masuda-Katsuse, I.; De Cheveigne, A.: Restructuring speech representations using a pitch-adaptive time–frequency smoothing and an instantaneous-frequency-based f0 extraction: possible role of a repetitive structure in sounds1. *Speech Commun.*, **27** (3–4) (1999), 187–207.

[11] Moulines, E.; Charpentier, F.: Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech Commun.*, **9** (5–6) (1990), 453–467.

[12] Stylianou, Y.; Laroche, J.; Moulines, E.: High-quality speech modification based on a harmonic+ noise model, in *Fourth European Conf. on Speech Communication and Technology*, 1995.

[13] Stylianou, Y.; Cappé, O.; Moulines, E.: Continuous probabilistic transform for voice conversion. *IEEE Trans. Speech Audio Process.*, **6** (2) (1998), 131–142.

[14] Toda, T. *et al.*: The voice conversion challenge 2016. *Proc. of the Annual Conf. of the Int. Speech Communication Association, INTERSPEECH*, vol. 08-12-Sept, 2016, 1632–1636.

[15] Helander, E.; Virtanen, T.: Voice conversion using partial least squares regression. *Audio, Speech, Language Process., IEEE Trans.*, **18** (5) (2010), 912–921.

[16] Abe, M.; Nakamura, S.; Shikano, K.; Kuwabara, H.: Voice conversion through vector quantization. *ICASSP-88., Int. Conf. Acoust., Speech, Signal Process.*, **2** (1988), 655–658.

[17] Kain, A.; Macon, M.W.: Spectral voice conversion for text-to-speech synthesis, in *Acoustics, Speech and Signal Processing, 1998. Proc. of the 1998 IEEE Int. Conf. on*, vol. 1. IEEE, 1998, 285–288.

[18] Toda, T.; Black, A.W.; Tokuda, K.: Voice conversion based on maximum-likelihood estimation of spectral parameter trajectory. *IEEE Trans. Audio, Speech, Language Process.*, **15** (8) (2007), 2222–2235.

[19] Kim, E.-K.; Lee, S.; Oh, Y.-H.: Hidden markov model based voice conversion using dynamic characteristics of speaker, in *Fifth European Conf. on Speech Communication and Technology*, 1997.

[20] Zhang, M.; Tao, J.; Nurminen, J.; Tian, J.; Wang, X.: Phoneme cluster based state mapping for text-independent voice conversion, in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE Int. Conf. on*. IEEE, 2009, 4281–4284.

[21] Song, P.; Bao, Y.; Zhao, L.; Zou, C.: Voice conversion using support vector regression. *Electron. Lett.*, **47** (18) (2011), 1045–1046.

[22] Narendranath, M.; Murthy, H.A.; Rajendran, S.; Yegnanarayana, B.: Transformation of formants for voice conversion using artificial neural networks. *Speech Commun.*, **16** (2) (1995), 207–216.

https://doi.org/10.1017/ATSIP.2018.23 Published online by Cambridge University Press

[23] Desai, S.; Black, A.W.; Yegnanarayana, B.; Prahallad, K.: Spectral mapping using artificial neural networks for voice conversion. *IEEE Trans. Audio, Speech, Language Process.*, **18** (5) (2010), 954–964.

[24] Chen, L.-H.; Ling, Z.-H.; Liu, L.-J.; Dai, L.-R.: Voice conversion using deep neural networks with layer-wise generative training. *IEEE/ACM Trans. Audio, Speech, Language Process. (TASLP)*, **22** (12) (2014), 1859–1872.

[25] Van Den Oord, A. *et al.*: Wavenet: a generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[26] Erro, D.; Moreno, A.; Bonafonte, A.: Voice conversion based on weighted frequency warping. *IEEE Trans. Audio, Speech, Language Process.*, **18** (5) (2010), 922–931.

[27] Godoy, E.; Rosec, O.; Chonavel, T.: Voice conversion using dynamic frequency warping with amplitude scaling, for parallel or nonparallel corpora. *IEEE Trans. Audio, Speech, Language Process.*, **20** (4) (2012), 1313–1323.

[28] Erro, D.; Navas, E.; Hernaez, I.: Parametric voice conversion based on bilinear frequency warping plus amplitude scaling. *IEEE Trans. Audio, Speech, Language Process.*, **21** (3) (2013), 556–566.

[29] Wu, Z.; Virtanen, T.; Kinnunen, T.; Chng, E.S.; Li, H.: Exemplar-based voice conversion using non-negative spectrogram deconvolution, in *Eighth ISCA Workshop on Speech Synthesis*, 2013.

[30] Wu, Z.; Virtanen, T.; Chng, E.S.; Li, H.: Exemplar-based sparse representation with residual compensation for voice conversion. *IEEE/ACM Trans. Audio, Speech, Language Process.*, **22** (10) (2014), 1506–1521.

[31] Xu, N.; Tang, Y.; Bao, J.; Jiang, A.; Liu, X.; Yang, Z.: Voice conversion based on Gaussian processes by coherent and asymmetric training with limited training data. *Speech. Commun.*, **58** (2014), 124–138.

[32] Uriz, A.J.; Agüero, P.D.; Bonafonte, A.; Tulli, J.C.: Voice conversion using k-histograms and frame selection, in *Tenth Annual Conf. of the Int. Speech Communication Association*, 2009.

[33] Erro, D.; Polyakova, T.; Moreno, A.: On combining statistical methods and frequency warping for high-quality voice conversion. *ICASSP, IEEE Int. Conf. on Acoustics, Speech and Signal Processing - Proceedings*, 2008, 4665–4668.

[34] Nakashika, T.; Takashima, R.; Takiguchi, T.; Ariki, Y.: Voice conversion in high-order eigen space using deep belief nets, in *Interspeech*, 2013, 369–372.

[35] Nguyen, H.Q.; Lee, S.W.; Tian, X.; Dong, M.; Chng, E.S.: High quality voice conversion using prosodic and high-resolution spectral features. *Multimed. Tools. Appl.*, **75** (9) (2016), 5265–5285.

[36] Erro, D.; Moreno, A.: Weighted frequency warping for voice conversion. *Proc. of the Annual Conf. of the International Speech Communication Association, INTERSPEECH*, **2** (2007), 1465–1468.

[37] Toda, T.; Muramatsu, T.; Banno, H.: Implementation of computationally efficient real-time voice conversion, in *Thirteenth Annual Conf. of the Int. Speech Communication Association*, 2012.

[38] Wang, H.: A mobile world made of functions. *APSIPA Trans. Signal Inf. Process.*, **6** (2017), e2.

[39] Erro, D.; Moreno, A.; Bonafonte, A.: Flexible harmonic/stochastic speech synthesis, in *SSW*, 2007, 194–199.

[40] Depalle, P.; Helie, T.: Extraction of spectral peak parameters using a short-time fourier transform modeling and no sidelobe windows, in *Applications of Signal Processing to Audio and Acoustics, 1997. 1997 IEEE ASSP Workshop on*. IEEE, 1997, 4–pp.

[41] Erro, D.; Moreno, A.: A pitch-asynchronous simple method for speech synthesis by diphone concatenation using the deterministic plus stochastic model, in *Proc. SPECOM*, 2005.

[42] Paliwal, K.K.: Interpolation properties of linear prediction parametric representations, in *Fourth European Conf. on Speech Communication and Technology*, 1995.

[43] El-Jaroudi, A.; Makhoul, J.: Discrete all-pole modeling. *IEEE Trans. Signal Process.*, **39** (2) (1991), 411–423.

[44] Kalva, H.; Colic, A.; Garcia, A.; Furht, B.: Parallel programming for multimedia applications. *Multimedia Tools Appl.*, **51** (2) (2011), 801–818.

[45] Asanovic, K. *et al.*: The landscape of parallel computing research: A view from Berkeley. Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, Tech. Rep., 2006.

[46] Corrales-Garcia, A.; Martinez, J.L.; Fernandez-Escribano, G.; Quiles, F.J.: Energy efficient low-cost video communications. *IEEE Trans. Consum. Electron.*, **58** (2) (2012), 513–521.

[47] Kwedlo, W.: A parallel em algorithm for Gaussian mixture models implemented on a numa system using openmp, in *Parallel, Distributed and Network-Based Processing (PDP), 2014 22nd Euromicro Int. Conf. on* IEEE, 2014, 292–298.

[48] Mitra, G.; Johnston, B.; Rendell, A.P.; McCreath, E.; Zhou, J.: Use of simd vector operations to accelerate application code performance on low-powered arm and intel platforms, in *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th Int.* IEEE, 2013, 1107–1116.

[49] Ha, S.-W.; Park, H.-C.; Han, T.-D.: Mobile digital image stabilisation using simd data path. *Electron. Lett.*, **48** (15) (2012), 922–924.

[50] Dagum, L.; Menon, R.: Openmp: an industry standard api for shared-memory programming. *IEEE Comput. Sci. Eng.*, **5** (1) (1998), 46–55.

[51] Williams, A.: C++ Concurrency in Action. Manning Publications, Shelter Island, New York, USA, 2017.

[52] Guennebaud, G. *et al.*: Eigen v3. http://eigen.tuxfamily.org, 2010.

[53] Berndt, D.J.; Clifford, J.: Using dynamic time warping to find patterns in time series, in *KDD workshop*, vol. 10, no. 16. Seattle, WA, 1994, 359–370.

[54] Hill, M.D.; Marty, M.R.: Amdahl's Law in the Multicore Era. *Computer*, **41** (7) (2008), 33–38.

[55] Yang, S.; Xie, L.; Chen, X.; Lou, X.; Huang, D.; Li, H.: Statistical parametric speech synthesis using generative adversarial networks under a multi-task learning framework. *arXiv preprint arXiv:1707.01670*, 2017.

[56] Toda, T.; Ohtani, Y.; Shikano, K.: One-to-many and many-to-one voice conversion based on eigenvoices. 2007.

[57] Saito, D.; Yamamoto, K.; Minematsu, N.; Hirose, K.: One-to-many voice conversion based on tensor representation of speaker space, in *Twelfth Annual Conf. of the Int. Speech Communication Association*, 2011.

[58] Muramatsu, T.; Ohtani, Y.; Toda, T.; Saruwatari, H.; Shikano, K.: Lowdelay voice conversion based on maximum likelihood estimation of spectral parameter trajectory, in *Proc. 2008 Autumn Meeting of Acoustic Society of Japan*, 2008, 3–4.

Shuhua Gao received a Bachelor degree from Shanghai Jiao Tong University in 2012 and a Master degree from Beihang University in 2015, both majoring in automation. He is currently pursuing the Ph.D. degree in National University of Singapore. His main research interests include the development and application of machine learning and evolutionary computation.

https://doi.org/10.1017/ATSIP.2018.23 Published online by Cambridge University Press

Xiaoling Wu received a Bachelor degree from National University of Singapore in 2017. She is currently a software engineer.

Cheng Xiang received the B.S. degree in mechanical engineering from Fudan University, China in 1991; M.S. degree in mechanical engineering from the Institute of Mechanics, Chinese Academy of Sciences in 1994; and M.S. and Ph.D. degrees in electrical engineering from Yale University in 1995 and 2000, respectively. He is an Associate Professor in the Department of Electrical and Computer Engineering at the National University of Singapore. His research interests include computational intelligence, adaptive systems and pattern recognition.

Dongyan Huang received her B.S. degree and M.S. degree from Xi'an Jiaotong University in electrical engineering. She has a Ph.D. degree in communication & electronics from Conservatoire National des Arts et Metiers. She is currently a senior research scientist & principal investigator at Institute for Infocomm Research (I^2R), A*STAR in Singapore. Her research interests include speech synthesis, voice conversion and transformation, emotional conversation generation and Human–Machine interaction.

https://doi.org/10.1017/ATSIP.2018.23 Published online by Cambridge University Press