Voichap: a standalone real-time voice change application on iOS platform

Xiaoling Wu[1], *Shuhua Gao*[1], *Dong-Yan Huang*[2], *Cheng Xiang*[1]

[1]Department of Electrical&Computer Engineering, National University of Singapore, Singapore
[2]Human Language Technology Department, Institute for Infocomm Research/A*STAR, Singapore
{a0115281,shuhua_gao}@u.nus.edu, huang@i2r.a-star.edu.sg, elexc@nus.edu.sg

Abstract

High-quality voice mimicry is appealing to everyone. However, only few vocal geniuses are endowed with the talent for vivid mimicry. Professional mimics have to be trained and practice over many years for various vocal skills, such as vocal control, precision in pitch, sense of rhythm and personal style, etc. To help achieve our dream for fascinating voice mimicry, such as speaking in a celebrity's voice, we have developed a real-time voice conversion technology for the general users. You can specify any target (like your friend or a celebrity) for your voice conversion as long as the target's training utterances are available. To facilitate easy use, we have implemented it efficiently as a mobile application on the iOS platform, called *Voichap*, which can generate a desired natural target voice. Notably, the complete training and conversion process is performed locally in a reasonable time, with no need for on-line server service, to improve the user experience. Just three steps are enough to use this application: choose a target, record your voice and then have fun listening to your converted voice.

Index Terms: real-time voice conversion, mobile application, harmonic plus stochastic model (HSM), parallel computing

1. Introduction

Voice conversion means the modification of a source speaker's voice to make it sound like the voice of another speaker (target speaker) while keeping the linguistic information unchanged. There are many potential applications of voice conversion, for example, we can equip text-to-speech (TTS) systems with voice conversion capacity to provide various distinctive voice choices to the user[1]. Some other scenarios for voice conversion may include voice recognition, voice pathology, video games and other entertainment applications. For example, a single actor can dub for all roles in a computer animated film by converting her/his voice to any roles voice, which will save the labor cost and maximize the profits.

The main underlying mechanism of voice conversion technology is to modify the nonlinguistic information such as the voice characteristics while leaving the linguistic information unchanged. In literature, various voice conversion approaches have been proposed with emphasis on different features, rules and requirements, such as the initial mapping code books [2], weighted frequency warping [3], Gaussian Mixture Model (GMM) [4] and artificial neural networks [5]. In recent years, as the increasing computation power and larger amount of data become available, more and more deep learning techniques have been adopted for voice conversion, for instance, the deep neural networks with layer-wise generative training [6] and the deep belief nets [7]. Remarkably, the DeepMind team has proposed a generative model for raw audios, called *WaveNet*, and they claim that this model can generate speech which mimics any human voice and which sounds more natural than the best existing Text-to-Speech systems, reducing the gap with human performance by over 50% [8]. However, though many of these algorithms may produce high-quality converted voice in the naturalness and similarity sense [9], it is still questionable whether they can be implemented effectively on a mobile platform, such as iOS or Android, due to the limited computation resource.

We first made a search on Apple store and Google play for present applications labeled with *voice change/conversion*. It was found that most of the existing voice conversion applications only deal with the modification of spectral features to mimic robot-like voices for fun, such as *VoiceLab* and *Voice Changer*. Only few applications are designed to mimic another predefined person's voice, such as Trump Voice Changer, which can speak the input text in a Trump-like voice, and the *Celebrity Voice Changer Lite*, which provides a fixed list of celebrities as your voice conversion target. However, according to the user reviews, there are mainly three limitations for these existent applications: first, the produced utterance sounds unnatural, dissimilar to the target's voice or even hard to understand; second, they need Internet connection to their on-line servers, which may be inconvenient and has potential issues on user privacy; last, no application supports the addition of custom new target speakers.

To this end, the main purpose of our study is to develop a stand-alone mobile application for voice conversion. This local application should generate converted voice for any specified target speaker, which sounds natural and similar, and allows the user to add custom target speakers. Therefore, both training and conversion need to be done in a single mobile phone within an acceptable time. This objective demands highly efficient implementation of voice conversion algorithms on mobile platforms. After an extensive comparison, the harmonic plus stochastic model (HSM) approach is chosen for the underlying conversion algorithm due to its satisfactory conversion performance and potentially high efficiency [10]. Then we implement this algorithm with the C++ programming language and advanced numerical computation optimizations. An iOS application is afterwards developed based on this algorithm and real device tests show that our implementation technology is efficient enough for high-quality voice conversion in a reasonable time: a 10-second utterance takes about 1.5 seconds for conversion. To the best of our knowledge, *Voichap* is the first public mobile application for full-featured human voice conversion: you can add new custom targets and specify any target to convert your voice to.

The remaining of this paper is organized as follows. In Section 2, the theoretical foundation of the underlying voice conversion algorithm is briefly introduced. In Section 3, the implementation techniques for numerical algorithm acceleration are presented. Then, Section 4 focuses on the development and design patterns of the iOS application *Voichap*. The final application product and its test are demonstrated in Section 5. Finally, concluding remarks followed by our future intentions for

further improvement are presented in Section 6.

2. Overview of the HSM-based voice conversion algorithm

Nowadays, artificial neural networks, especially deep learning, are quite popular and it seems they can solve questions in many fields, from computer vision to natural language processing and even unmanned vehicle drive. As aforementioned, the famous research institute on deep learning, DeepMind, has developed the *WaveNet* for human speech synthesis. However, we must realize that deep learning is not a panacea. The two essential requirements of deep learning, huge data and high computational capacity, will limit its application in many cases, especially on resource-limited mobile platforms. For a mobile application, it would be crazy to ask the user for a huge amount of training audios. What's worse, the poor GPU power on most mobile phones can hardly accelerate the neural network training process. Therefore, for this project, we try to implement a traditional voice conversion algorithm based on the harmonic plus stochastic model (HSM) [10]. Though the focus of our study is on the efficient implementation of this algorithm and the development of an iOS application, for completeness, the basic principle of the algorithm is briefly introduced in the following.

The overall idea of HSM is to represent the speech signal as a sum of various harmonically related sinusoids with time-varying parameters and a noise-like component. The harmonic component is only present in the voiced speech segments and is decomposed into a number of sinusoidal harmonics. Then the non-sinusoidal signal components, which may be caused by the frication or breathing noise, are modeled by the stochastic component. The voice conversion system based on HSM models is divided into two parts, training and conversion, which is illustrated in Figure 1. In the training phase, a conversion model is generated based on a set of source and target voice inputs. Then in the conversion phase, using the acquired conversion model, the system can convert the voice of a source speaker to that of the target speaker by modifying the corresponding voice features. For more details, interested readers are referred to [10, 11].

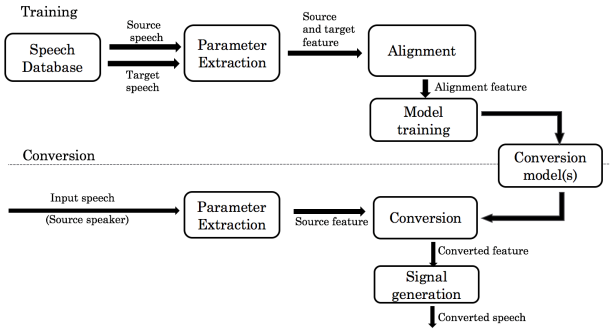


Figure 1: *Overview of the HSM voice conversion algorithm.*

In our application, the input source speeches and target speeches can be directly recorded by the phone, following which a binary format model file is generated by the training part. After we get the model, the user (source speaker) can speak anything to the phone, which will be collected by the recorder, and the conversion part will modify the source voice features according to the model to make it sound like the target speaker voice. This overall work flow of our voice change application, *Voichap*, is shown in Figure 2. In the following sections, the techniques for efficient implementation and the application graphical user interface development will be detailed.

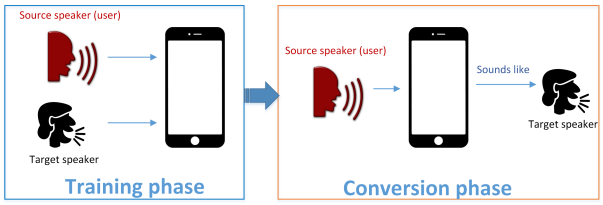


Figure 2: *Workflow of the voice change application.*

3. Highly-efficient implementation of the voice change algorithms

Generally speaking, hardware is always faster than software. Thus, we need to make full use of the hardware characteristics for efficient computation-intensive algorithm implementation, such as the data-level parallelism and core-level concurrency supported by most modern CPUs. To better utilize such hardware facilities, a relatively low-level but high-performance programming language is the best candidate for implementation. For the iOS platform, the main programming languages are the classic Objective-C and the emerging Swift. However, both languages are inconvenient for low-level API interaction and lack a versatile linear algebra library. Besides, though Apple recommends Swift for iOS application development, it is quite awkward to manipulate data memory directly with Swift due to its automatic memory management. Therefore, to gain best algorithm performance and to facilitate the iOS application development, we choose C++, a native language, for the core voice conversion algorithm implementation, and Swift for the user interface building.

Since Swift does not support direct interoperability with C++, we first wrap the input/output interfaces of the C++ algorithm with C language, which can communicate with Swift later. Besides, almost all modern operation systems have a C++ compiler, which means our algorithm code implemented in C++ can be easily ported to multiple platforms, including mobile platforms such as iOS and Android. The programing paradigm with C++ implemented core algorithm for multiple-platform support is summarized in Figure 3. Two numeric computation techniques have been adopted to accelerate the voice conversion algorithm, including of vectorization and multiple-core parallel computing.

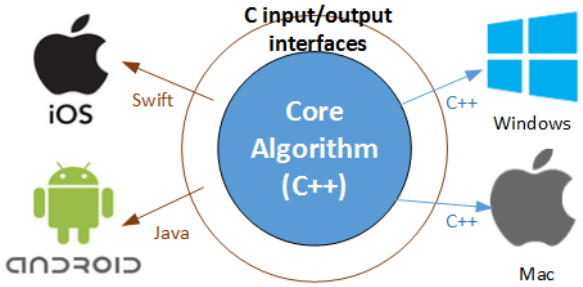


Figure 3: *C++ implementation of the voice conversion algorithm for multiple platform support.*

3.1. Vectorization with SIMD instructions

Single instruction for multiple data (SIMD), is a class of parallel computers in Flynns taxonomy [12]. It is the process of rewriting a loop so that instead of processing a single element of an array N times, it processes multiple of elements of the array simultaneously, called vectorization. iPhone 7 is equipped with an Apple A10 CPU including of two 64-bit 2.34 GHz ARMv8-A cores. Since ARMv8 provides an SIMD instruction set, NEON, for efficient processes of multimedia process, we use the *Eigen* library to accelerate our code performance by vectorization. *Eigen* is a C++ template library for linear algebra computations, which supports explicit vectorization using SSE, AVX and NEON instruction sets. Besides, compared with MATLAB, which is an interpreted language, C++ is a powerful native language and the *Eigen* library can make use of the SIMD instructions of modern processors, both of which contribute to the much faster speed of C++ programs with *Eigen*. In converting MATLAB prototype code to C++, *Eigen* can also be used to replace some MATALB built-in functions for matrix manipulation, such as the *eig* and *sum* functions, to make the C++ program more efficient and more understandable.

3.2. Multiple-core parallel computing

Parallel computing is a type of computation in which many calculations or the execution of process are carried out simultaneously on multiple cores. Open Multi-Processing(OpenMP) and Grand Central Dispatch (GCD) are used on Windows and Mac/iOS platforms respectively to boost the code performance. OpenMP is an application programming interface (API) for multiprocessing programing using shared memory in C or C++ supported on most platforms. GCD is a counterpart technology developed by Apple Inc specifically for iOS/Mac systems with multi-core processors. In practice, our algorithm is first developed and tested with the powerful Visual Studio IDE on Windows 10. Afterwards, it is ported to macOS/iOS with only few necessary changes, mainly replacing OpenMP with GCD support, thanks to the high platform-independent capacity of C++.

3.3. Test results

To visually demonstrate the power of the above techniques in the speeding up of the numerical algorithm implementation, the algorithm performance is tested under various environment configurations. Figure 4 shows the running time of the algorithms training phase with four different configurations on PC with Windows 10 and Inter Core i7 of 4 hardware cores. Since there are multiple training audio samples and their feature extractions are independent, we can allocate them on different cores for parallel processing. As we can see, utilizing both vectorization and multi-core parallel programming is the most efficient among the 4 different configurations. Therefore, in the subsequent iOS application development, well adopt these two techniques to speed up the conversion algorithm.

4. iOS application development

4.1. Overview of the functional modules

In software engineering, it is well known that modularization is a necessary method for large-scale software development. In this project, the mobile application is also constructed with various modules, with each module responsible for its own function. The module structure of this application is illustrated in Figure 5, which includes six functional modules.

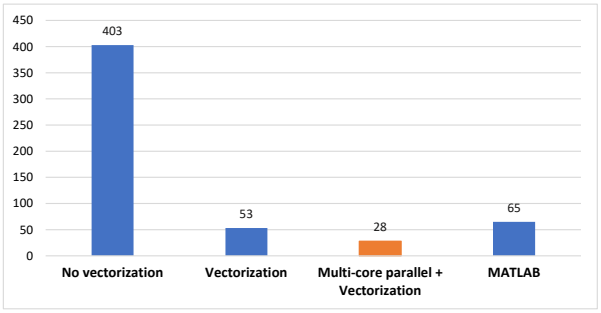


Figure 4: *Running time of the training phase with different configurations(unit: second). Left to right: C++ program: C++ program with vectorization: C++ program with both vectorization and multi-core parallel computing; MATLAB program.*

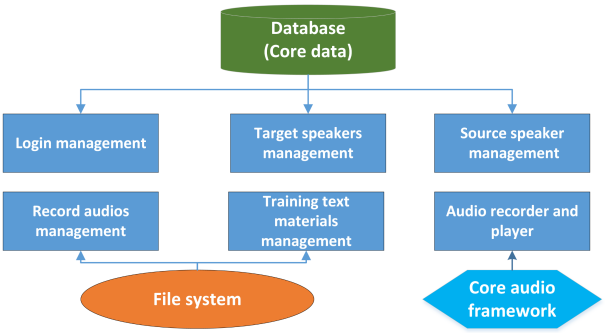


Figure 5: *Functional module structure of the iOS application.*

4.2. Some specific design patterns

Developing a well-performing iOS application is a challenging task, which involves UI design, business logic modeling, coding with Swift/Objective-C and debugging in an iterative manner. In the following, two typical design considerations are presented as representatives of the whole process and the coding/debugging details are neglected here for conciseness.

4.2.1. MVC design pattern for the user interface

We use the Model-View-Control (MVC) pattern for our user interface, which is illustrated in Figure 6. The benefits for using this pattern are that many components in this application will be more reusable following the *single responsibility principle* in software engineering [13]. By adopting the MVC design pattern, applications tend to be more manageable and more easily extensible. The three objects in the MVC framework have different roles and can interact with each other: user action in the view layer that creates or modifies the data informs the controller to update its model; on the other hand, when a model is updated, it notifies a controller object to refresh the view with corresponding changes.

4.2.2. Data-driven design and presentation in collection views

To display multiple items of record audios and target speakers in better organization, the table view and collection view are extensively used in the GUI design of our application. For more

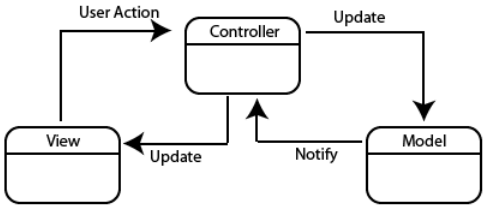


Figure 6: *Model-View-Control(MVC) design pattern.*

flexibility and reusability of the program, the best practice is to separate the data from its visualization through a *UI-Data-Operation* style. In the iOS development framework, the data is stored in a data source object and the operation is represented by a *delegate* object.

The single responsibility of the data source object is to provide data and it does not care about how the data will be displayed. To customize the rendering of data, i.e., the specific appearance of data on the UI, we can adopt a delegate object. In addition, the UI displaying the data is synchronized with the underlying data source through the delegate object. After all these connections are set up, we can simply make modifications on the data source, like insertion or deletion, and the corresponding table/collection views will be updated automatically. Therefore, it is exactly the data that drives the evolution of the dynamic view. More importantly, delegation is ideal for passing information between objects in order to change the behavior of certain objects, which is actually an implementation of the classic *observer pattern* in a one-to-one sense [14]. With this design pattern, we can better decouple the data source and its presentation.

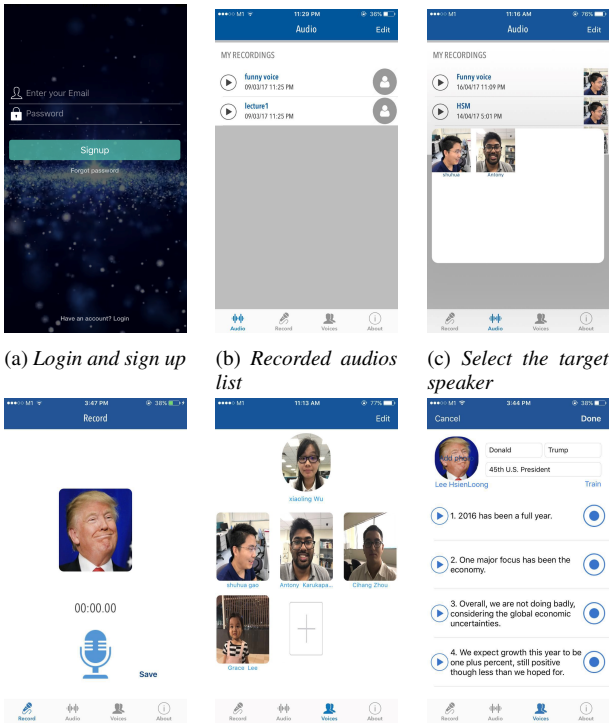
5. Demonstration of the application

In this section, we will introduce the functions of our iOS application *Voichap* which has been deployed on an iPhone 7 device, whose screenshots are exhibited in Figure 7.

To better explain the user guides for our *Voichap*, we show an example about how to convert your voice to President Trump's voice in Figure 8. First, you create a new target in the target management screen in Figure 7e. Then by clicking your portrait or Trump's portrait, the information and training window in Figure 7f will show up. In this window, the personal information can be edited and the training audio samples can be loaded here (your voice can be recorded directly while Trump's may be downloaded from *YouTube*). It should be noted that you and Trump have to speak the same sentences for training purpose. After the 20 training samples from both you and Trump have been fed, just click the *Train* button to start training, which may take about 40 seconds. From then on, you can enjoy your words "spoken" by Trump in the *Speak here* screen in Figure 7d: first choose Trump as the target; then hold the microphone button for speaking. Once the button is released, you'll hear the same speech as you have just said in a short moment, but in Trump's voice, which is generated by our underlying voice conversion system in real-time manner.

6. Conclusions

In this study, a HSM model based human voice change algorithm is implemented quite efficiently with C++ by adopting advanced numerical computation techniques including of vectorization and multi-core parallel computing. The fast speed and satisfactory voice conversion quality of this implementation is proved through a well-designed iOS application. Extensive experiments have been conducted to demonstrate this result: on the iPhone 7 platform, the training phase for a 20 samples dataset takes about 40s and the conversion of a 10s input audio needs about 1.2s. Besides, the converted voice sounds very natural and similar to the target's voice in most cases.



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

(d) *Speak here* (e) *User and targets management* (f) *User/target information and training*

Figure 7: *User interface of the Voichap application*

Since the current application can only convert a speakers voice to another persons voice, that is, in a *speech-to-speech* manner, our future work will focus on the integration of this technique into TTS (text-to-speech) systems such that texts can be read with a specified speaker's voice directly. To sum up, for the first time our study has proved the feasibility of full-featured custom voice change on a mobile platform using only the local computation resource through the highly efficient algorithm implementation with modern and advanced techniques.

7. Acknowledgments

The authors would like to acknowledge Dr. Daniel Erro, the original designer of the conversion algorithm, whose website can be found at https://aholab.ehu.es/users/derro/home.html.

8. References

[1] A. Kain and M. W. Macon, "Spectral voice conversion for text-to-speech synthesis," in *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, vol. 1. IEEE, 1998, pp. 285–288.

[2] M. Abe, S. Nakamura, K. Shikano, and H. Kuwabara, "Voice conversion through vector quantization," in *Acoustics, Speech, and*

Figure 8: *A typical usage scenario of Voichap: how to make yourself sound like President Trump?*

Signal Processing, 1988. ICASSP-88., 1988 International Conference on. IEEE, 1988, pp. 655–658.

[3] D. Erro, A. Moreno, and A. Bonafonte, "Voice conversion based on weighted frequency warping," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 5, pp. 922–931, 2010.

[4] Y. Chen, M. Chu, E. Chang, J. Liu, and R. Liu, "Voice conversion with smoothed gmm and map adaptation." in *INTERSPEECH*, 2003.

[5] S. Desai, A. W. Black, B. Yegnanarayana, and K. Prahallad, "Spectral mapping using artificial neural networks for voice conversion," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 5, pp. 954–964, 2010.

[6] L.-H. Chen, Z.-H. Ling, L.-J. Liu, and L.-R. Dai, "Voice conversion using deep neural networks with layer-wise generative training," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 22, no. 12, pp. 1859–1872, 2014.

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

[8] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *CoRR abs/1609.03499*, 2016.

[9] M. Wester, Z. Wu, and J. Yamagishi, "Analysis of the voice conversion challenge 2016 evaluation results," *submitted to) Interspeech*, 2016.

[10] E. Banos, D. Erro, A. Bonafonte, and A. Moreno, "Flexible harmonic/stochastic modeling for hmm-based speech synthesis," *Proc. V Jornadas en Tecnologias del Habla*, pp. 145–148, 2008.

[11] D. E. Eslava and A. M. Bilbao, "Intra-lingual and cross-lingual voice conversion using harmonic plus stochastic models," *Barcelona, Spain: PhD Thesis, Universitat Politechnica de Catalunya*, 2008.

[12] R. Duncan, "A survey of parallel computer architectures," *Computer*, vol. 23, no. 2, pp. 5–16, 1990.

[13] R. C. Martin, *Agile software development: principles, patterns, and practices*. Prentice Hall, 2002.

[14] J. Hannemann and G. Kiczales, "Design pattern implementation in java and aspectj," in *ACM Sigplan Notices*, vol. 37, no. 11. ACM, 2002, pp. 161–173.