REAL-TIME SINGING VOICE CONVERSION PLUG-IN

Shahan Nercessian, Russell McClellan, Cory Goldsmith, Alex M. Fink, and Nicholas LaPenn

iZotope, Inc.
Boston, MA, USA
{shahan|rmcclellan|cgoldsmith|afink|nlapenn}@izotope.com

ABSTRACT

In this paper, we propose an approach to real-time singing voice conversion and outline its development as a plug-in suitable for streaming use in a digital audio workstation. In order to simultaneously ensure pitch preservation and reduce the computational complexity of the overall system, we adopt a source-filter methodology and consider a vocoder-free paradigm for modeling the conversion task. In this case, the source is extracted and altered using more traditional DSP techniques, while the filter is determined using a deep neural network. The latter can be trained in an end-to-end fashion and additionally uses adversarial training to improve system fidelity. Careful design allows the system to scale naturally to sampling rates higher than the neural filter model sampling rate, outputting full-band signals while avoiding the need for resampling. Accordingly, the resulting system, when operating at 44.1 kHz, incurs under 60 ms of latency and operates 20 times faster than real-time on a standard laptop CPU.

1. INTRODUCTION

Singing voice conversion (SVC) is an audio style transfer application which converts the voice of a sung performance to that of another without changing its underlying content or melody [1]. It can be used for expressive and creative voice manipulations that go beyond conventional effects. Relative to voice conversion applied to speech, SVC has stronger demands on accurate pitch preservation as humans are sensitive to pitch instabilities in singing [2].

SVC has been dominated by deep learning approaches of late. With some exceptions, methods attempt to predict converted acoustic features, and use vocoders to synthesize waveforms from said features [3]. The end-to-end adversarial SVC (EA-SVC) method inverts a learned latent representation with a MelGAN [4] generator, using adversarial training to improve signal plausibility [5]. DiffSVC uses a diffusion model to improve acoustic feature modeling [6]. As state-of-the-art neural vocoders often lack sufficient pitch stability, FastSVC [1] conditions waveform generation on a harmonic excitation signal. Most approaches focus on fidelity improvements, with less attention placed on their deployability as real-time streaming plug-ins that operate seamlessly in conventional audio workflows.

In our own previous works, we have considered different feature representations and end-to-end training mechanisms [2, 3]. Most recently [7], we explored a vocoder-free alternative [8] that was implemented in an end-to-end context using a variation of a WORLD feature representation [9]. In this case, we achieved SVC by processing the input signal rather than synthesizing a new one based on it, effectively guaranteeing its relative pitch contour.

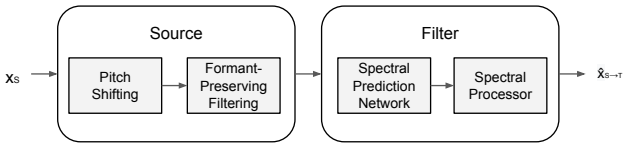

Figure 1: *Proposed RT-SVC system block diagram.*

In this work, we propose a real-time SVC (RT-SVC) approach and implement it as a real-time plug-in. Drawing from [7], we model SVC using a source-filter method, combining pitch-shifting techniques with a lightweight, low-latency neural filter model. We highlight design choices to balance fidelity and performance, and means of scaling our methods to different sampling rates. Our paper is organized as follows: Section 2 describes the proposed method, Section 3 discusses its realization as a real-time plug-in, Section 4 reports experimental findings, and Section 5 draws conclusions.

2. PROPOSED METHOD

2.1. System overview

We consider source and target vocalists S and T, respectively. Our aim is to determine a suitable waveform $x_{S\to T}$ capturing the performance (content) of an input source waveform x_S, while assuming the character of T (style). The conversion task involves two transformations on x_S, as illustrated conceptually in Figure 1. The first stage pitch shifts the input by a constant factor, such that the resulting $x_{S,PS}$ is reflective of the register of the vocalist T. The second stage applies linear time-varying (LTV) filtering to the pitch-shifted result so that the timbre of the result conceivably matches that of the vocalist T. This is modeled using a deep learning model trained in an end-to-end manner over a dataset of recordings of the vocalist T. As stated, the problem naturally lends itself to a source-filter approach.

While many SVC approaches depend on neural vocoders to synthesize new waveforms from inferred representations, we approach the task by processing the input signal. In the context of a real-time system, this offers several advantages, including:

- *Pitch preservation*: Parametric/neural vocoders are prone to pitch errors, and cannot ensure relative pitch input contour preservation in their outputs. Meanwhile, we preserve pitch exactly, barring absolute shifts which we can reliably apply.
- *Reduced complexity*: Removing the need for a vocoder reduces computational footprint and latency in our system.

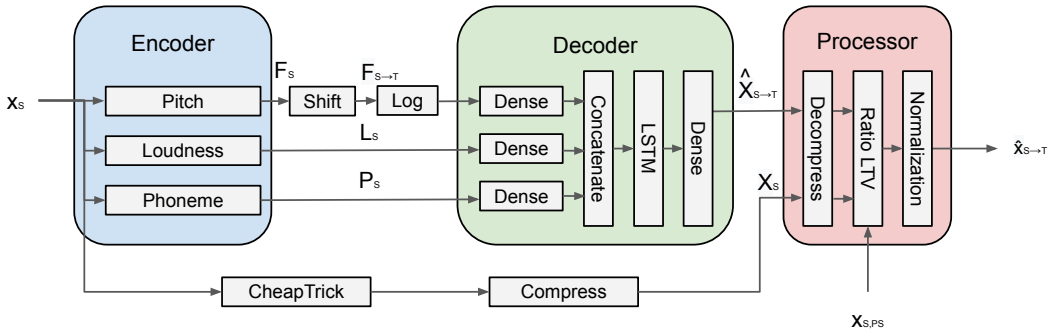

Figure 2: *Fully differentiable end-to-end neural filter model block diagram.*

- *Extension to arbitrary sampling rates*: We can extend our method to sampling rates beyond that used in training, producing wide-band outputs while only needing to model the SVC effect over a perceptually relevant sub-band [10].
- *Voice interpolation*: We can interpolate between source and modeled timbres via a convex combination of acoustic features, with perfect recovery of the source timbre if desired.

2.2. Vocoder-free design with WORLD log Mel spectrograms

Defining $\mathcal{F}$ and $\mathcal{F}^{-1}$ as the forward and inverse short-time Fourier transforms (STFT), respectively, we describe our approach as a spectral process. At a nominal sampling rate $f_{s,0} = 22.05$ kHz, we use hop and frame lengths of $H = 256$ and $N = 1024$ samples, respectively. A signal x has the source-filter decomposition

$$x = \mathcal{F}^{-1}\left[\mathcal{F}(e) \odot \mathcal{F}(h)\right] \tag{1}$$

where e and h denote the time-varying source (i.e. excitation) and filter function of x, respectively. Proper estimation of the spectral envelope $|\mathcal{F}(h)|^2$ leads to extraction of a predominantly flat excitation spectrum. To this end, we leverage the CheapTrick algorithm from WORLD analysis [9], which estimates spectral envelopes $sp = \left|\mathcal{F}(\hat{h})\right|^2$ via a fundamental frequency (f_0) dependent smoothing of signal power spectra.

We seek to determine $\mathcal{F}(e_{S\to T})$ and $\mathcal{F}(h_{S\to T})$ given x_S, from which we can compute $x_{S\to T}$ according to equation (1). To model the excitation spectrum, we apply a pitch shift to x_S to register match it against the target vocalist, resulting in $x_{S,PS,0}$. Robust formant preservation during pitch shifting is a system requirement, and we propose a generic means for providing this for any pitch shifting process through *formant-preserving post-filtering*. Leveraging CheapTrick again, we derive the spectrum of what would be the post-filtered, pitch-shifted signal $x_{S,PS}$ as

$$\mathcal{F}(x_{S,PS}) = \sqrt{\frac{sp_S}{sp_{S,PS,0}}} \odot \mathcal{F}(x_{S,PS,0}) \tag{2}$$

where sp_S and $sp_{S,PS,0}$ are estimates of the spectral envelopes of x_S and $x_{S,PS,0}$, respectively.

To reduce the dimensionality of features ultimately predicted by our deep learning model, we use a compressed representation called the WORLD log Mel spectrogram [7], given by

$$X = \log_{10}(\mathbf{M}\sqrt{sp} + \epsilon) \tag{3}$$

where $\mathbf{M}$ is the Mel basis matrix used to compute an M-band Mel spectrogram from an N-point STFT, and $\epsilon = 10^{-5}$ is used for numerical stability. It is similar to the generalized Mel cepstrum [3], except that it makes for a more obvious differentiable implementation to support end-to-end training. We use $M = 80$ in this work. A decompressed approximation of X is then

$$sp^{\dagger} = \left[\mathbf{M}_0^{\dagger}(10^X - \epsilon)\right]^2 \tag{4}$$

where $\mathbf{M}_0^{\dagger} = \max(\mathbf{M}^{\dagger}, 0)$ and $\mathbf{M}^{\dagger}$ denotes the pseudo-inverse of $\mathbf{M}$. Given $sp_S^{\dagger}$ derived from the source WORLD log Mel spectrogram X_S, the estimated excitation spectrum is given by

$$\mathcal{F}(\hat{e}_{S\to T}) = \sqrt{\frac{1}{sp_S^{\dagger}}} \odot \mathcal{F}(x_{S,PS}) \tag{5}$$

We task a deep learning model (discussed in Section 2.3) to provide estimates for $\hat{X}_{S\to T}$, resulting in $\mathcal{F}(\hat{h}_{S\to T}) = \hat{sp}_{S\to T}^{\dagger}$ via equation 4. Combining into equations (1) and (5), the converted waveform is estimated as

$$\hat{x}_{S\to T} = \mathcal{F}^{-1}\left[\kappa \odot \sqrt{\frac{\hat{sp}_{S\to T}^{\dagger}}{sp_S^{\dagger}}} \odot \sqrt{\frac{sp_S}{sp_{x_{S,PS,0}}}} \odot \mathcal{F}(x_{S,PS,0})\right] \tag{6}$$

where κ is a normalization term computed empirically at each time step to ensure that the modified spectrum L1 norm matches that of $\mathcal{F}(x_S)$. Equation 6 contains four distinct parts: 1) pitch shifting, 2) formant-preserving post-filtering, 3) timbral modification, and 4) normalization. The resulting ratio-based LTV filter is non-negative by design, mitigating potential phase coherence issues.

2.3. Neural filter model

The goal of the neural filter model, outlined in Figure 2, is primarily to infer WORLD log Mel spectrograms $\hat{X}_{S\to T}$ to match the timbre of the target singer while maintaining the content of the source. The LTV filtering outlined in Section 2.2 can be made fully differentiable given access to the various spectral envelopes and waveforms as input, so it is implemented as part of our model in order to enable its end-to-end training [11], yielding $\hat{x}_{S\to T}$.

To perform SVC for any source S, we must create a singer-independent encoding for x_S. Our encoder extracts source loudness L_S deterministically, using the frame-level root-mean-square (RMS) converted to decibels during training (with system hop and frame lengths). We also use the tonality-gated f_0 contour F_S extracted using DIO [9] within WORLD analysis during training. At

inference time, the pitch feature is offset based on the pitch shift that we apply, yielding the target pitch contour $F_{S\to T}$. Lastly, to capture linguistic content, we extract a phonetic posteriorgram P_S [5] from a phoneme classifier. We found it instructive to pre-emphasize the input prior to passing it to the phoneme classifier using a finite impulse response filter of the form

$$y[t] = k_0 x[t] - k_1 x[t-1] \tag{7}$$

with $k_0 = 1.0$ and $k_1 = 0.97$. The classifier then passes 40 Mel frequency cepstral coefficients (MFCCs) extracted from the pre-emphasized signal through a unidirectional recurrent architecture consisting of two long short-term memory (LSTM) layers with 256 units, and a final dense layer yielding a 61-dimensional output vector of phoneme class probabilities at each time step. The network is trained on the TIMIT dataset [12].

The decoder builds upon a lightweight, real-time variant of the architecture in [11], where each multi-layer perceptron (MLP) head uses a single dense layer with 256 units, layer normalization and ReLU nonlinearity. As the initial architecture only considered frequency and loudness features, we add an additional head for P_S. In doing so, we effectively model the way in which the timbre of a phonetic sequence of a target vocalist is varied based on changes in delivery (e.g. when belting a high note loudly). The inputs and outputs of each encoding head are combined and fed through a single 256-unit LSTM layer, followed by a dense layer which outputs a WORLD log Mel spectrogram. The end-to-end audio processor contained within the model filters input audio using the decompressed spectral envelopes, resulting in the model output waveform.

2.4. Training objective

The neural filter model is trained as an autoencoder, and therefore, we have $x_S = x_{S,PS,0} = x_{S,PS} = x_T = x_{S\to T}$, $X_S = X_T = X_{S\to T}$, $F_S = F_T = F_{S\to T}$, etc. during training. In this sense, the model is trained end-to-end, but admittedly, is never exposed to pitch-shifted audio (or any of its associated audio artifacts), as the training objective is merely one of self-reconstruction.

Model training minimizes a combination of conventional negative log likelihood loss terms ensuring good average fidelity and adversarial loss terms promoting plausible system outputs as determined by a discriminator network [4]. To this end, we considered the time-domain multi-scale discriminator architecture in [4] and a single-scale version of the spectral domain architecture as in [13]. Also similar to [13], we actually observed better performance using a spectral domain discriminator for the task. Given a suitably trained and frozen (phonetic) encoder, the full objective function for the neural filter model is

$$\mathcal{L} = \mathcal{L}_{MSE} + \mathcal{L}_{MSL} + \mu\mathcal{L}_G \tag{8}$$

where $\mathcal{L}_{MSE}$ is the mean squared error (MSE) defined on WORLD Mel spectrograms (i.e. the decoder outputs), $\mathcal{L}_{MSL}$ is the end-to-end multi-spectrogram loss (MSL) [11], $\mathcal{L}_G$ is the end-to-end adversarial generator hinge loss, and μ is a hyperparameter set to 0.5 in this work. As in [14], we noticed that the usual deep feature matching loss associated with [4] tended to slow down convergence, and that $\mathcal{L}_{MSE} + \mathcal{L}_{MSL}$ was sufficient for stabilizing adversarial training. The discriminator is trained to minimize its corresponding end-to-end discriminator hinge loss.

3. REAL-TIME PLUG-IN IMPLEMENTATION

We have implemented our system as a real-time plug-in in C++. Here, we outline practical considerations for such a realization.

3.1. Streaming feature extraction

We approximate the frame-based loudness feature used during training, computing RMS in a zero-latency fashion via a 1-pole infinite impulse response (IIR) filter with a time constant equal to the system stride. We replace DIO with our proprietary low-latency pitch detection algorithm, and use our implementation of the Lent algorithm [15] as a real-time pitch shifter. Loudness and pitch features are sampled according to the system stride so that they are aligned to their frame-based counterparts. Lastly, we refactor the CheapTrick C++ implementation [9] to handle buffered audio streams.

3.2. Extension to higher sampling rates

We design the plug-in to extend processing to a sampling rate $f_s = G \cdot f_{s,0}$ ($G \geq 1$) without the need for resampling. We consider how our feature representations vary as a function of G, and devise schemes to roughly neutralize this effect, so as not to create a large input feature mismatch from training. Our pitch detection/shifting algorithms and the loudness and pitch feature computations in Section 3.1 are sample-rate agnostic by construction. STFT frame and hop lengths scale with G (while ensuring N to be an even power of 2), and as most fast Fourier transform (FFT) implementations are unnormalized, we carefully scale power and magnitude spectra (as used in CheapTrick or to generate MFCCs) by $1/G^2$ and $1/G$, respectively. We construct Mel bases and pseudo-inverse matrices at f_s while maintaining their respective lower and upper frequency bounds at $f_{s,0}$. We roughly preserve the response of the pre-emphasis filter in equation (7) using generalized filter coefficients $k_0 = G$, $k_1 = G - 1 + k_{1,0}$, $k_{1,0} = 0.97$.

Lastly, we found that we only need to model SVC up to around 10 kHz (arguably lower) to yield a convincing effect, and that we can safely extend the LTV filter gain derived near this boundary to frequencies above it. This effectively amounts to injecting a properly scaled version of $\mathcal{F}(x_{S,PS})$ at frequencies above 10 kHz. This way, we produce wide-band SVC results, even though our network is only trained to model a smaller bandwidth arguably considered too narrow for music production purposes.

3.3. Model export

According to its gains in [16], we use TFLite as a real-time deep learning inference engine. To do so, we recreate encoder/decoder architectures in TensorFlow, explicitly defining a single time step of acoustic features as model input/output. We add LSTM state vectors as additional inputs/outputs so that we can propagate them between time steps and reset them as needed in the plugin. Lastly, we convert the resulting TensorFlow model to the TFLite format.

3.4. Plug-in performance

We incur about 45 ms of latency due to windowing in CheapTrick and the forward/inverse STFTs, and less than 15 ms of residual latency due to pitch detection/shifting, resulting in a total plug-in latency of just under 60 ms. The plug-in runs 20 times faster than real-time on a standard laptop CPU and can be launched from a conventional digital audio workstation.

Table 1: *Quantitative and qualitative model comparisons.*

Model	L_1	# Params.	Pitch shifter	MOS
Offline [7]	**0.042**	25.3M	Lent	3.99
			Phase Vocoder	**4.52**
RT-SVC	0.135	**2.19M**	Lent	3.37
			Phase Vocoder	3.89
RT-SVC$_{GAN}$	0.225	**2.19M**	Lent	*3.69*
			Phase Vocoder	4.20

4. EXPERIMENTAL RESULTS

We exemplify our methods using an internal collection of voice data. The dataset features recordings from 15 different singers, with approximately 2 hours of data for each singer. We consider 3 SVC models for each vocalist: our offline model used in [7], as well as RT-SVC models trained with and without adversarial loss terms (RT-SVC and RT-SVC$_{GAN}$, respectively). All systems are trained at 22.05 kHz, using 2-second audio clips and a batch size of 4. We use the Adam optimizer with a learning rate of 10^{-4} and train for 500,000 training steps. When leveraging adversarial training, we train the generator for 50,000 steps before training the discriminator. For subjective listening, we refer readers to our demo website at https://sites.google.com/izotope.com/rtsvc-demo.

Table 1 reports L_1 reconstruction error of log Mel spectrograms computed between inferred waveforms and their targets and mean opinion scores (MOS) collected from participants within our organization, across models trained on one of said singers. For the latter, participants were asked to rate examples from 0 to 100, and responses were rescaled to the conventional 1 to 5 scale. Overall, 11 people with proficient listening and varied musical experience evaluated our models. Indeed, our offline model outperforms RT-SVC models in terms of fidelity quantitatively and qualitatively. It tends to reconstruct prolonged vowels more consistently, and generally exhibits less leakage of the input speaker identity. For added perspective, we note that regardless of the neural filter model, the use of our proprietary high-latency phase vocoder pitch shifting algorithm can noticeably and universally affect fidelity as well, experiencing fewer audio artifacts across consonants and vowels relative to our Lent implementation. Nonetheless, our real-time model is over 10 times smaller, and when paired with the Lent pitch shifter, is amenable to streaming applications. Lastly, while RT-SVC$_{GAN}$ achieves worse average L_1 performance than RT-SVC, it produces more plausible outputs, as per its higher MOS score (see supplemental figures on our website for details).

5. CONCLUSIONS

We proposed an approach for real-time SVC and implemented it as a streaming plug-in. The method combines pitch shifting of the input signal and timbral transformation provided by a deep learning model. The novelty of the method is that it acts directly on the input signal instead of synthesizing a new waveform, reducing complexity and preserving the pitch of the original signal. As such, the model can extend to sampling rates beyond the nominal rate used by its deep learning model component. In future work, we are interested in improving excitation signal modeling, and specifically, to see if it is possible to inject the pitch shifting algorithm "in-the-loop" during training to improve fidelity. We would also like to improve acoustic feature/filter modeling for the real-time case, potentially leveraging recent advances on integrating streaming convolutional layers and related architectures [17].

6. REFERENCES

[1] S. Liu et al., "FastSVC: Fast cross-domain singing voice conversion with feature-wise linear modulation," in *IEEE Int. Conf. on Mul. and Ex. (ICME)*, 2021, pp. 1–6.

[2] S. Nercessian, "Zero-shot singing voice conversion," in *Int. Soc. for Mus. Inf. Ret. Conf. (ISMIR)*, 2020, p. 70–76.

[3] S. Nercessian, "End-to-end zero-shot voice conversion using a DDSP vocoder," in *IEEE Work. on App. of Sig. Proc. to Aud. and Ac. (WASPAA)*, 2021, pp. 1–5.

[4] K. Kumar et al., "MelGAN: Generative adversarial networks for conditional waveform synthesis," in *Adv. in Neu. Inf. Proc. Sys. (NeurIPS)*, 2019, vol. 32.

[5] H. Guo et al., "Phonetic posteriorgrams based many-to-many singing voice conversion via adversarial training," *arXiv:2012.01837*, 2020.

[6] S. Liu, Y. Cao, D. Su, and H. Meng, "DiffSVC: A diffusion probabilistic model for singing voice conversion," *arXiv:2105.13871*, 2021.

[7] S. Nercessian, "Differentiable WORLD synthesizer-based neural vocoder with application to end-to-end audio style transfer," in *154th Aud. Eng. Soc. Conv. (AES)*, 2023.

[8] J.W. Kim, H.Y. Jung, and M. Lee, "Vocoder-free end-to-end voice conversion with transformer network," in *IEEE Int. Joint Conf. on Neu. Net. (IJCNN)*, 2020, pp. 1–8.

[9] M. Morise, F. Yokomori, and K. Ozawa, "WORLD: a vocoder-based high-quality speech synthesis system for real-time applications," *IEICE Trans. on Inf. and Sys.*, vol. E99-D, no. 7, pp. 1877–1884, 2016.

[10] T. Saeki, Y. Saito, S. Takamichi, and H. Saruwatari, "Real-time, full-band, online DNN-based voice conversion system using a single CPU," in *Interspeech*, 2020, pp. 1021–1022.

[11] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," in *Int. Conf. on Learn. Rep. (ICLR)*, 2020, pp. 26–30.

[12] J. S. Garapolo et al., *TIMIT Acoustic-Phonetic Continuous Speech Corpus LDC93S1*, Linguistic Data Consortium, Philadelphia, 1993.

[13] A. Wright, V. Välimäki, and L. Juvela, "Adversarial guitar amplifier modelling with unpaired data," in *IEEE Int. Conf. on Ac., Speech and Sig. Proc. (ICASSP)*, 2023.

[14] R. Yamamoto, E. Song, , and Jae-Min Kim, "Parallel WaveGAN: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram," in *IEEE Int. Conf. on Ac., Speech and Sig. Proc. (ICASSP)*, 2020, p. 6199–6203.

[15] K. Lent, "An efficient method for pitch shifting digitally sampled sounds," *Computer Music Journal*, vol. 13, no. 4, pp. 65–71, 1989.

[16] D. Stefani, S. Peroni, and L. Turchet, "A comparison of deep learning inference engines for embedded real-time audio classification," in *Int. Conf. on Dig. Aud. Eff. (DAFx)*, 2022, pp. 256–263.

[17] A. Caillon and P. Esling, "Streamable neural audio synthesis with non-causal convolutions," in *Int. Conf. on Dig. Aud. Eff. (DAFx)*, 2022, pp. 320–327.