VOICE CONVERSION MATLAB TOOLBOX

David Sündermann

Siemens Corporate Technology, Munich, Germany
Technical University of Catalonia, Barcelona, Spain
`david@suendermann.com`

ABSTRACT

This paper documents function and properties of the Voice Conversion Matlab Toolbox (version 2007-02-18). It contains information on system requirements, an overview about the modules included, shows examples of applying the toolbox to voice conversion based on vocal tract length normalization (VTLN) and linear transformation in a step-by-step manner, and gives details about the parameter settings.

Index Terms— voice conversion, Matlab, toolbox

1. INTRODUCTION

Voice conversion is the transformation of the voice characteristics of a source towards a target voice [1].

The Voice Conversion Matlab Toolbox described in this paper is a collection of Matlab scripts that enables the user to rapidly design, modify, and test voice conversion algorithms based on

- VTLN in frequency domain [2] or time domain [3],
- linear transformation [4].

It contains several signal processing tools, which are the fundamentals of the aforementioned voice conversion techniques, such as

- a pitch tracker,
- a voicing detector,
- a dynamic programming module,
- dynamic time and frequency warping algorithms,
- a program for monitoring and manually modifying pitch marks,
- feature conversion tools for linear predictive coefficients, line spectral frequencies, (mel frequency) cepstral coefficients, residual coefficients according to Ye and Young [5] and Sündermann [6], and sinusoidal coefficients,
- interpolation tools for linear, cubic spline, mel-scale, and two-dimensional interpolation,
- Gaussian mixture modeling,
- hashing,
- k-means clustering,
- least squares fitting,
- linear transformation,
- VTLN,
- objective error measures as line spectral distortion, signal-to-noise ratio, residual distance [6], and mahalanobis distance,
- pitch-synchronous overlap and add,
- residual prediction,
- unit selection,
- vector smoothing based on normal distribitions (also two-dimensional),
- text-independent speech alignment.

This paper does not intend to discuss all tools contained in this toolbox (there are more than 200), since they are documented in the header of the respective source files, which can be easily accessed by typing

```
help tool
```

at the Matlab prompt, where `tool` has to be replaced by the respective tool's name such as `vtln`.

2. SYSTEM REQUIREMENTS

The toolbox was tested on Matlab version 6.5 (Release 13) on a Windows XP platform and on Matlab version 6.1 (Release 12.1) on a Linux platform[1]. Both systems had the signal

[1]Most of the delivered scripts are in DOS-formatted ASCII code. If used under Linux, they should be converted using the `dos2unix` command to avoid data format errors.

processing as well as the statistics toolbox installed, but the author tried to avoid using them to make the toolbox applicable to a wider range of systems.

In addition to the Matlab functionality of the toolbox, there are a few algorithms based on other environments or programs such as

- Perl version 5.8,
- the author's Language Resources Generation Toolbox [7],
- Praat [8] version 4.4,
- Cygwin version 2.510.

3. VTLN-BASED VOICE CONVERSION

3.1. Pitch Tracking and Voicing Detection

The speech processing of this toolbox is mainly based on the pitch-synchronous paradigm, which requires the speech data to be pitch-tracked. The pitch tracker described in [9] comes along with the toolbox, however, according to the author's experience and a recent pitch tracker evaluation [10], it achieves a rather poor performance leading to a considerable number of artifacts. The Praat program produces much more reliable pitch marks and is to be used as standard pitch tracker in the following.

In the directory of the Matlab toolbox (in the following referred to as *toolbox directory*), where we expect all Matlab and other commands to be executed (unless otherwise specified), is a folder `data` that contains a number of example speech files: `f.01.wav` to `f.10.wav` of a female voice and `m.01.wav` to `m.10.wav` of a male voice. We move to this folder by typing at the Cygwin prompt

```
cd data
```

Now, the list of all `wav` files in the current folder is taken, a Praat script is generated and finally executed. Here, scripts from the Language Generation Toolbox, and Praat's command line version `praatcon`[2] are used[3].

```
ls *.wav | replaceStr.pl ˆ 'bash ...
..\/wav2pp.bash ' > wav2pp.full.bash
bash wav2pp.full.bash > wav2pp.praat
praatcon wav2pp.praat
```

The result are `pp`, i.e. PointProcess files, the Praat format for pitch marks, one for each `wav` file available.

So far, the pitch marks are only determined in voiced signal parts. Since required for the following speech processing, also (pseudo) pitch marks for the unvoiced signal portions are generated by applying a pitch mark interpolation according to [11]. In doing so, the voicing information is to be preserved, as it will play an important role for the voice conversion. The internal pitch mark format is that according to [9], which stores a vector of the pitch period lengths in numbers of samples, i.e. integer numbers, into a `pm` file. Furthermore, the respective voicing information is stored into a parallel `v` (voicing) file. At first, the header of the `pp` files are removed resulting in files of the type `pit`

```
ls *.pp | replaceStr.pl '.pp$' > stem.txt
cat stem.txt | replaceStr.pl ˆ ...
'tail +7 ' | paste.pl '.pp > ' '' | ...
paste.pl stem.txt '' | ...
paste.pl '.pit' '' > pp2pit.bash
bash pp2pit.bash
```

Now, a Matlab batch processing script is generated that transforms the `pit` file to the `pm` type and also produces the `v` files

```
cat stem.txt | ...
replaceStr.pl ˆ "pit2pm('data\/" | ...
paste.pl ".pit', 'data/" '' | ...
paste.pl stem.txt '' | ...
paste.pl ".wav', 'data/" '' | ...
paste.pl stem.txt '' | ...
paste.pl ".pm', 'data/" '' | ...
paste.pl stem.txt '' | ...
paste.pl ".v');" '' > ../tmp_pit2pm.m[4]
```

Now, we type at the Matlab prompt (in the toolbox directory)

```
tmp_pit2pm
```

producing the desired pitch mark and voicing files.

3.2. Estimating the Warping Factor and Fundamental Frequency Ratio

According to the definition of frequency as well as time domain VTLN as given by [12], above all, two parameters are required for the VTLN-based voice conversion: the warping factor `alpha` and the fundamental frequency ratio `rho`, cf. [13]. This section shows, how these parameters are estimated on the training data.

To begin with, so called file list files (cf. `fileListFile2file`) are generated that contain the names of `wav`, `pm`, and `v` files by typing at the Cygwin prompt (in the toolbox directory)

[2]Under Linux, Praat's command line version is `praat`.

[3]The character sequence '`...`' is to indicate that the line is continued.

[4]Usually, all working files, data, temporary files, etc. should be located in the `data` directory. Exceptions are the temporary files generated by several Matlab tools (cf. `getRandFile` and `clrTemp`). All other files that for some reasons have to be (temporarily) located in the toolbox directory, should be preceded by the prefix `tmp_` to make them easily detectable.

```
ls data/f*.wav > data/f.wav.l
ls data/f*.pm > data/f.pm.l
ls data/f*.v > data/f.v.l
ls data/m*.wav > data/m.wav.l
ls data/m*.pm > data/m.pm.l
ls data/m*.v > data/m.v.l
```

Now, the announced parameters are trained on the training data, for instance for female-to-male conversion, typing at the Matlab prompt

```
[alpha, rho] = getWarpingFactor( ...
'data/f.wav.l', 'data/f.pm.l', ...
'data/m.wav.l', 'data/m.pm.l')
```

3.3. Frequency Domain and Time Domain VTLN

Now, the objective is to convert a source speech file by means of the estimated warping parameters towards the target voice. This is done by means of the Matlab command

```
vtlnBasedVc('data/f.01.wav', ...
'data/f.01.pm', 'data/f.01.v', ...
'data/outputVtln.wav', alpha, rho, 'fdvtln')
```

storing the converted speech as `data/outputVtln.wav`. At this point, and also in the following, we apply the trained parameters to files that also served as training material. This is only for reasons of convenience. If the reader has doubts of the applicability of the trained parameters to unseen data, he is invited to split the data into training and testing data by himself.

The above command line is the frequency domain VTLN version. For the time domain version, replace the last argument by `'tdvtln'`.

It shall be mentioned that the parameter estimation discussed in Section 3.2 via `getWarpingFactor` is based on frequency domain VTLN. The particular warping function can be defined as an additional argument (consult `help getWarpingFactor`), default is the symmetric piece-wise function according to [14]. Since the warping factor strongly depends on the warping function used, this function must also be specified as additional argument to `vtlnBasedVc`, if different from the default. When time domain VTLN is applied (which usually produces a superior speech quality), there is no choice as for the warping function, since it corresponds to the symmetric piece-wise function by definition; for a proof, see [3].

In a recent study [12], the author showed that by altering the mentioned parameters, `alpha` and `rho`, he is able to produce several (at least five) well-distinguishable voices based on one source voice. This is to invite the user of this toolbox to manually play with these parameters to achieve the best desired effect. It shall also be mentioned that, in terms of the objective criterion used for training (cf. [2]), the automatically determined parameter settings might be optimal; perceptively, however, this may not be the case.

4. VOICE CONVERSION BASED ON LINEAR TRANSFORMATION

The linear transformation task is much more complex than the VTLN-based one. Therefore and, in particular, due to the number of parameters involved, we make use of a config file, that is located in the `data` folder and is called `config.txt`. For the syntax of this config file consult `help getParameter`, a function that reads parameters out of a config file.

For the parameters that will be used by the following operations, have a look at Table 1.

4.1. Parameter Training

The training is performed by means of the command

```
trainVc('data/config.txt')
```

All trained parameters are stored to the `vcParamFile`, cf. Table 1. One can always access the contents of this file by using `file2x`, e.g. by typing

```
vcParam = file2x(getParameter( ...
'vcParamFile', 'data/config.txt'))
```

In particular, the field `vcParam.general` is of interest, since it contains all parameters specified in the config file and other ones derived in the training.

4.2. Conversion

The conversion is performed using the command

```
vc('data/config.txt')
```

Again, all necessary settings have to be applied to the config file.

5. COPYRIGHT

All scripts of the Voice Conversion Matlab Toolbox are written by the author in the years 2003 to 2007. Table 2 contains the exceptions.

6. REFERENCES

[1] D. Sündermann, "Voice Conversion: State-of-the-Art and Future Work," in *Proc. of the DAGA*, Munich, Germany, 2005.

[2] D. Sündermann, H. Ney, and H. Höge, "VTLN-Based Cross-Language Voice Conversion," in *Proc. of the ASRU*, Virgin Islands, USA, 2003.

[3] D. Sündermann, A. Bonafonte, H. Ney, and H. Höge, "Time Domain Vocal Tract Length Normalization," in *Proc. of the ISSPIT*, Rome, Italy, 2004.

[4] Y. Stylianou, O. Cappé, and E. Moulines, "Statistical Methods for Voice Quality Transformation," in *Proc. of the Eurospeech*, Madrid, Spain, 1995.

[5] H. Ye and S. Young, "High Quality Voice Morphing," in *Proc. of the ICASSP*, Montreal, Canada, 2004.

[6] D. Sündermann, A. Bonafonte, H. Ney, and H. Höge, "A Study on Residual Prediction Techniques for Voice Conversion," in *Proc. of the ICASSP*, Philadelphia, USA, 2005.

[7] D. Sündermann, "A Language Resources Generation Toolbox for Speech Synthesis," in *Proc. of the AST*, Maribor, Slovenia, 2004.

[8] P. Boersma, "Praat, a System for Doing Phonetics by Computer," *Glot International*, vol. 5, no. 9/10, 2001.

[9] V. Goncharoff and P. Gries, "An Algorithm for Accurately Marking Pitch Pulses in Speech Signals," in *Proc. of the SIP*, Las Vegas, USA, 1998.

[10] B. Kotnik, H. Höge, and Z. Kacic, "Evaluation of Pitch Detection Algorithms in Adverse Conditions," in *Proc. of the Speech Prosody*, Dresden, Germany, 2006.

[11] A. Black and K. Lenzo, *Building Synthetic Voices*, Carnegie Mellon University, Pittsburgh, USA, 2003.

[12] D. Sündermann, G. Strecha, A. Bonafonte, H. Höge, and H. Ney, "Evaluation of VTLN-Based Voice Conversion for Embedded Speech Synthesis," in *Proc. of the Interspeech*, Lisbon, Portugal, 2005.

[13] D. Sündermann, *Text-Independent Voice Conversion (Draft)*, Ph.D. thesis, Bundeswehr University Munich, Munich, Germany, 2006.

[14] L. Uebel and P. Woodland, "An Investigation into Vocal Tract Length Normalization," in *Proc. of the Eurospeech*, Budapest, Hungary, 1999.

[15] A. Kain and M. Macon, "Spectral Voice Conversion for Text-to-Speech Synthesis," in *Proc. of the ICASSP*, Seattle, USA, 1998.

[16] D. Sündermann, H. Höge, A. Bonafonte, H. Ney, A. Black, and S. Narayanan, "Text-Independent Voice Conversion Based on Unit Selection," in *Proc. of the ICASSP*, Toulouse, France, 2006.

[17] D. Sündermann, H. Höge, A. Bonafonte, H. Ney, and J. Hirschberg, "Text-Independent Cross-Language Voice Conversion," in *Proc. of the Interspeech*, Pittsburgh, USA, 2006.

[18] A. Kain, *High Resolution Voice Transformation*, Ph.D. thesis, Oregon Health and Science University, Portland, USA, 2001.

[19] D. Sündermann, H. Höge, A. Bonafonte, H. Ney, and A. Black, "Residual Prediction Based on Unit Selection," in *Proc. of the ASRU*, San Juan, Puerto Rico, 2005.

[20] D. Sündermann, H. Höge, and T. Fingscheidt, "Breaking a Paradox: Applying VTLN to Residuals," in *Proc. of the ITG*, Kiel, Germany, 2006.

[21] J. Chen and A. Gersho, "Real-Time Vector APC Speech Coding at 4800 bps with Adaptive Postfiltering," in *Proc. of the ICASSP*, Dallas, USA, 1987.

parameter	**description**	**example**
general		
`vcParamFile`	contains all parameters derived in training	`data/vc.mat`
`vcParamNarrow`	if the memory-consuming parts of `vcParamFile` can be removed; this is recommendable, if `residualPrediction` is switched off	`0`
training		
`normPitch`	f_n in Hz, cf. [13]	`100.0`
`sourceWavFileListFile`	`wav` and `pm` training file list files of source and target	`data/f.wav.l`
`sourcePmFileListFile`		`data/f.pm.l`
`targetWavFileListFile`		`data/m.wav.l`
`targetPmFileListFile`		`data/m.pm.l`
linear transformation		
`linearTransformationMode`	the type of linear transformation – either according to [4] (`stylianou`) or to [15] (`kain`)	`stylianou`
`covarianceType`	type of covariance matrices; for the possible types, cf. `trainGmm`	`diag`
`featNum`	dimensionality of features (D according to [13])	`32`
`mixNum`	number of Gaussian mixture densities	`4`
speech alignment, for details cf. [16]		
`twType`	speech alignment type; `dtw` for dynamic time warping or `tw` for text-independent alignment based on unit selection	`dtw`
`twDp`†	if dynamic programming (DP) is to be applied or a context-free minimization (i.e., $w = 1$ in the unit selection formula in [13]), which is much faster	`1`
`twFeatNum`†	dimensionality of features used for unit-selection-based alignment	`16`
`twAlpha`†	w according to [13]	`0.3`
`twDeltaM`†	cf. `twUnitSelectionCost`	`1`
`twNBest`†	cf. `fullDp`	`3`
parallelization; since text-independent speech alignment can be very time-consuming (see [17]), it should be parallelized		
`twParallelFileNum`	the index of the file to be processed in the training file list file – the resulting frame index sequence is written to `twParallelIndexFile`, and then the program stops. If all index files of all parallel computations have been generated, they have to be concatenated in their correct order as given by the file list files (e.g. by means of the Cygwin command `cat`) and written to an index file containing the alignment information of the whole corpus. Now, the config file entry `twParallelIndexFile` has to be adapted accordingly, and the parameter `twParallelFileNum` has to be set to `-1`, which means that the alignment is read from this file instead of computing it. If no parallelization is to be carried out, set `twParallelFileNum` to `0`.	`0`
`twParallelIndexFile`	see description of `twParallelFileNum`	`data/index.txt`

parameter	**description**	**example**
conversion		
`lsfSmoothingOrder`	The linearly transformed line spectral frequencies (LSFs) are smoothed by means of `smoothing` using this smoothing order; for a justification see [18].	`3.0`
`inputWavFile`	input `wav`, `pm`, `v` and output `wav` files or file list files	`data/f.01.wav`
`inputPmFile`		`data/f.01.pm`
`inputVoicingFile`		`data/f.01.v`
`outputWavFile`		`data/output.wav`
residual prediction		
`residualPrediction`	if residual prediction is to be applied	`0`
`linearTransformation`	if linear transformation is to be applied; if not, VTLN-based voice conversion is performed	`1`
`nResidual`‡	the number of residuals to be considered. A smaller number can significantly accelerate the voice conversion. Put `Inf` to consider all residuals seen in training.	`Inf`
`useInputPhase`‡	if the source phase spectra are to be copied to the target	`0`
`alpha1`‡	w_1 according to [19] ($w_2 = 1 - \omega_1$, $w_3 = 0$)	`0.0`
`suendermannSmoothOrder`‡	residual smoothing strength, σ_0 according to [13]	`2.0`
VTLN		
`alphaVoiced`⋆	VTLN warping factor for the residuals of voiced signal portions (referred to as `alpha` in Section 3.2), cf. [20]	`1.6`
`alphaUnvoiced`	VTLN warping factor for unvoiced signal portions; should be around 1.0 and only be varied, if the unvoiced sounds of both involved speakers strongly differ	`1.0`
synthesis		
`unvoicedAmp`	amplification factor of unvoiced signal portions; this can be of interest to adapt the gains of voiced and unvoiced signal portions, which are handled seperately in the conversion process and also behave differently from speaker to speaker	`1.0`
`lengthRatio`⋆	fundamental frequency ratio (referred to as `rho` in Section 3.2)	`0.6`
`perceptual`	if the perceptual filter according to [21] is to be used	`1`

Table 1. Parameters of the config file.

⋆ These parameters are estimated in training. Consequently, they should only be explicitly given in the config file, if the estimated values are to be ignored and replaced.

† Only applicable if `twType` is `tw`.

‡ Only applicable if `residualPrediction` is `1`.

file	author	reference	date
consist.m	I. Nabney	http://www.ncrg.aston.ac.uk/netlab/	1996–2001
dist2.m			
distchck.m	The MathWorks, Inc.	http://www.mathworks.com/	1993–2002
dp.m*	D. Ellis	dpwe@ee.columbia.edu	2003-03-15
dtw.m*			
dumpmemmex.dll	The MathWorks, Inc.	http://www.mathworks.com/	1993–2006
find_pmarks.m*	W. Goncharoff	goncharo@ece.uic.edu	1997-12-06
frq2mel.m	M. Brookes	mike.brookes@imperial.ac.uk	1998-04-03
getMfccfilterParams.m*	Interval Research Corp.	http://www.interval.com/	1998
gmmactiv.m*	I. Nabney	http://www.ncrg.aston.ac.uk/netlab/	1996–2001
gmm.m			
gmmactive.m*			
gmmem.m*			
gmminit.m*			
gmmpost.m			
gmmprob.m			
hashadd.m	D. Mochihashi	daichi.mochihashi@atr.jp	2004
hashinit.m			
hashval.m			
inpoly.m	D. Doolin	doolin@ce.berkeley.edu	1999-03-26
kmeans2.m*	I. Nabney	http://www.ncrg.aston.ac.uk/netlab/	1996–2001
mel2frq.m	M. Brookes	mike.brookes@imperial.ac.uk	1998-04-03
mfcc2spec.m*	Interval Research Corp.	http://www.interval.com/	1998
mvnpdf.m	The MathWorks, Inc.	http://www.mathworks.com/	1993–2002
normpdf.m			
simmx.m	D. Ellis	dpwe@ee.columbia.edu	2003-03-15
spec2mfcc.m*	Interval Research Corp.	http://www.interval.com/	1998

Table 2. Copyright notes.

* modified by the author of this paper