IDMT Time Stretch Pitch Shift Library Documentation

06.01.2016

Fraunhofer IDMT
Hanna Lukashevich, lkh@idmt.fraunhofer.de
Sascha Grollmisch, goh@idmt.fraunhofer.de
Jakob Abeßer, abr@idmt.fraunhofer.de

Contents

1 Introduction

The Time Streching and Pitch Shifting library by Fraunhofer IDMT enabled the pitching while keeping the timing of audio material. In the same way the library can be used for stretching the audio material without changing its pitch. The pitch can be changed by -6 to +6 which covers one full octave. The tempo can be set from 50% to 200%.

1.1 Supported Platforms

The library can be used on

- Windows (Windows 7 or higher),
- Mac OS X (10.6 64 bit or higher),
- Linux (Fedora 20, 64 Bit or higher),
- Android (armeabi-v7a), and
- iOS.

2 General Usage

The audio samples are passed block-wise into the library and synthesized according to the parameters. The library returns the number of needed input samples for one output block. If time stretching is applied the number of input samples differs from the number of output samples.

2.1 Package Content

This package contains the library for different plattforms including the header and lib files. It also contains sample files that demonstrate the usage for desktop plattforms.

2.2 Usage under Windows / Linux / OSX / iOS

Include `TimeStretchPitchShiftIDMT_cAPI.h` in your project. Under Windows / iOS, link the static parts of the library. Under Windows / Linux / OSX, copy the dynamic library (`dll` or `dylib`) to your binary folder. The iOS library is compiled to be used in the simulator and on the device.

2.3 Interface Description under Windows / Linux / OSX / iOS

The class has to be initialized by calling `createTimeStretchPitchShiftIDMT()`. Set pitch shifting via `setPitchShiftSemitones(semitones)`. Set time stretching via `setTimeStrechFactor(factor)` where 2.0 means half tempo and 0.5 double tempo. After setting the parameters check the expected input and output blocksizes with `getInputBlockSize()` and `getOutputBlockSize()`. Then process the audio data blockwise by calling `processSamples(const float* input, float* output)` with the expected input and output sizes. Since internal buffering is used for better audio quality call `reset()` for reseting the internal buffers when completely new audio is processed. If errors occur during processing the library returns values unequal to zero and the last error message can be received by calling `getLastErrorMessage()`. Finally, the allocated memory from the library can be freed by calling `deleteTimeStretchPitchShiftIDMT()`.

2.4 Usage under Android

Copy `armeabi-v7a` to libs folder in your Android project and the `de` folder to sources. The library only runs on the device itself not in the simulator!

2.5 Interface Description under Android

Create library instance via `createTimeStretchingPitchShift()` and release memory when finished by calling `deleteTimeStretchingPitchShift()`. Process audio samples in short format (`AudioFormat.ENCODING_PCM_16BIT`) to algorithm by calling `processSampleBuff(short[] buffer)`. Get the expected input and output sizes by calling `getInputBlockSize()` and `getOutputBlockSize()`. Set pitch shifting via `setPitchShiftSemitones(semitones)`. Set time stretching via `setTimeStrechFactor(factor)` where 2.0 means half tempo and 0.5 double tempo. Since internal buffering is used for better audio quality call `reset()` for reseting the internal buffers when completely new audio is processed.

2.6 Samples

2.6.1 Windows/Linux/OSX

The sample folder contains the file `main.cpp` that shows how pitch shift wave files by using the library. Pitch shift can be set via command line arguments. The input data is processed blockwise like it could be done in real time use case. For easy compilation, the file `CMakeLists.txt` is included. It allows to create `make`, `Visual Studio`, and `XCode` files.
Under Linux and OSX, go to the sample folder and execute `cmake .` in the commandline which will crate make files. Then, compile via `make TimeStretchPitchShiftSample`. An executable with same name will be created.
Under Linux, add the current directory to `LD_LIBRARY_PATH` so `libtime_stretching_idmt_shared.so` can be found.
Under Windows, run CMake with desired generator (Visual Studio, NMake, etc.) and compile `TimeStretchPitchShiftSample`. Copy `time_stretching_idmt_shared.dll` to the executable folder.
For further information, see `www.cmake.org`.