The Audio Degradation Toolbox and its Application to Robustness Evaluation

Mauch, M; Ewert, S

THE AUDIO DEGRADATION TOOLBOX AND ITS APPLICATION TO ROBUSTNESS EVALUATION

Matthias Mauch **Sebastian Ewert**
Queen Mary University of London, Centre for Digital Music
{matthias.mauch, sebastian.ewert}@eecs.qmul.ac.uk

ABSTRACT

We introduce the Audio Degradation Toolbox (ADT) for the controlled degradation of audio signals, and propose its usage as a means of evaluating and comparing the robustness of audio processing algorithms. Music recordings encountered in practical applications are subject to varied, sometimes unpredictable degradation. For example, audio is degraded by low-quality microphones, noisy recording environments, MP3 compression, dynamic compression in broadcasting or vinyl decay. In spite of this, no standard software for the degradation of audio exists, and music processing methods are usually evaluated against clean data. The ADT fills this gap by providing Matlab scripts that emulate a wide range of degradation types. We describe 14 degradation units, and how they can be chained to create more complex, 'real-world' degradations. The ADT also provides functionality to adjust existing ground-truth, correcting for temporal distortions introduced by degradation. Using four different music informatics tasks, we show that performance strongly depends on the combination of method and degradation applied. We demonstrate that specific degradations can reduce or even reverse the performance difference between two competing methods. ADT source code, sounds, impulse responses and definitions are freely available for download.

1. INTRODUCTION

Advances in audio and music processing technology depend on the ability to evaluate the success of different methods against each other, and to show in which ways they behave differently. Consequently, better evaluation methods and the creation of new, bigger ground-truth data sets are topics of lively research and debate [1, 7]. Considerably less attention is usually devoted to the question of whether the processing methods are robust against degradations in audio quality. For example, the robustness of automatic speech recognition systems is often only tested by adding noise or a limited set of additive environmental sounds [6], or by filtering the audio [11], e.g. using the FaNT toolkit [1]. Similarly, in music informatics only a few studies consider insufficient or variable audio quality, exceptions including two studies on the influence of MP3 compression on genre classification [2] and onset detection [12], and a study of onset detection methods under reverberation [21].

Real-world applications often require robustness against a broad range of signal degradations, as the quality of audio recordings is often entirely unknown. For example, mobile music-processing apps suffer from low quality microphones in smartphones [3], in addition to noisy recording environments and unpredictable non-linear processing in the phones' DSP chips. Similarly, guaranteeing a consistent audio quality in a series of recordings is almost impossible in exploratory ethnomusicology [13, 18]. Audio effects applied at radio stations can lead to significant challenges for audio identification methods. Many such applications are likely to receive input audio of mixed levels of degradation. Therefore, studying the influence of various signal degradations on a given method can aid the detection of problems and the development of appropriate countermeasures.

In this paper, we describe the Audio Degradation Toolbox, which offers a well-defined way of applying a wide range of audio degradations, from basic sanity checks over simple parameter sweeps to highly-specific degradations imitating audio sourced from radio, smartphones or vinyl records. The breadth of degradations goes beyond the application of noise and MP3 degradations currently used in the literature. Using four typical music informatics tasks as an example, we show how the toolbox can be used to analyse the behaviour of methods under degradation, compare the performance of competing approaches in various scenarios, locate conceptual weaknesses in methods and develop suitable countermeasures. To facilitate integration into existing evaluation workflows, the toolbox provides means to adjust ground-truth annotations to compensate for temporal distortions resulting from degradations. The combination of audio and ground-truth transformation functionality makes the toolbox an easy-to-use instrument to boost the informative value of systematic evaluations.

The rest of the paper is organised as follows: Section 2 describes the components of the toolbox itself and how they work together. In Section 3 we examine the behaviour of several music informatics methods under audio degradation. Section 4 summarises and concludes the paper.

[1] http://dnt.kr.hs-niederrhein.de/download.html

```
audio_out = degradationUnit_addNoise(
      audio, samplingFreq)
```

a) only audio processing

```
parameter.noiseColor = 'brown';
[audio_out, timestamps_out] =
   degradationUnit_addNoise(audio, samplingFreq,
                            timestamps, parameter)
```

b) with timestamps and a parameter variable

Figure 1: Example calls to a degradation unit function.

2. THE AUDIO DEGRADATION TOOLBOX

The Audio Degradation Toolbox (ADT) consists of Matlab code for the controlled degradation of audio signals, and for the adaptation of ground-truth to the degraded audio. Additionally, some scripts are provided to facilitate batch processing. The source code is hosted on SoundSoftware.org [2] and is freely available under a GPL license. The ADT is based on two concepts: *(a) degradation unit*: a Matlab function with optional parameters that performs a simple audio (and ground-truth) transformation, *(b) degradation*: parametrisation of several degradation units into a chain. The remainder of this section details the rationale and implementation of the degradation units and degradations.

2.1 Degradation Units

The ADT contains 14 basic degradation units implemented as Matlab functions. All degradation units use the same calling conventions. Figure 1a illustrates the simplest case, in which a degradation unit is called with two arguments: an audio data matrix (`audio`) and a scalar providing the audio sampling frequency (`samplingFreq`); the function returns a matrix of degraded audio data (`audio_out`) with the same sampling frequency as the input audio.

Some degradations change the speed of the audio or warp time in other ways (see detailed descriptions, below) with the result that timestamps annotated with respect to the original audio will no longer be valid. In particular, ground-truth annotated on the original audio would be meaningless on such degraded audio. Therefore, all degradation units implement a timestamp transformation that maps the input timestamps to the timeline of the output audio. The transformed ground-truth can then be used for evaluation on degraded audio data. Figure 1b illustrates how the user can provide additional timestamps (`timestamps`) and receive an output variable containing adjusted timestamps (`timestamps_out`). For convenience, all degradation units have a default setting, but they will typically be called with customised parameter settings, as in Figure 1b. More on parameters in Section 2.2.

What follows are short descriptions of the individual degradation units contained in the ADT. Degradation units that introduce a time warping are marked with the letter W. More documentation can be found as comment headers in the respective source code files.

[2] `http://code.soundsoftware.ac.uk/projects/audio-degradation-toolbox`

Room Impulse Responses [20]: Great Hall, Classroom, Octagon. **Smartphone Impulse Responses** (recorded for the ADT: exponential sine sweep, inverse filtering): Google Nexus One microphone, Google Nexus One speaker. **Vinyl** (recorded for the ADT: inverse filtered sine sweep output from iZotope Vinyl 1.73b plugin): Vinyl Player 1960 (raw and smoothed)

(a) Impulse Responses

Pub Sound Environment [9]. **Vinyl:** recorded for the ADT using the iZotope Vinyl plugin.

(b) Sounds

Table 1: Data included in the ADT.

Add Noise. Adds artificial random noise of different 'colours' (brown, pink, white, blue, violet) to the signal at a specified signal-to noise ratio (SNR). Implemented as white noise plus filter.

Add Sound. Adds an arbitrary signal to a given signal at a specified SNR and takes care of sampling rate conversions (the ADT comes with two sounds, see Table 1b).

Attenuation. Makes the signal quieter by a specified number of decibels or a factor specified.

Aliasing. Purposeful violation of the sampling theorem: the signal is down-sampled to a specified sampling rate without lowpass filtering. The original sampling rate is restored using a regular resampling method (with filtering).

Clipping. Normalises the audio signal such that a specified number or percentage of samples is outside the interval $[-1, 1]$, and each resulting sample x is clipped to $\mathrm{sign}(x)$.

Delay (W). Pads the beginning of the input audio with a specified number of zero samples. The timestamps are delayed accordingly.

Dynamic Range Compression (W). Applies a signal-dependent normalisation to the audio signal, reducing the energy difference between soft and loud parts of the signal. The implementation is adapted from [22], allowing specification of several parameters including attack, release, delay and slope. As the delay parameter introduces a constant time delay, the output timestamps are adjusted accordingly.

Apply Impulse Response (W). Filters the signal using one of six natural impulse responses (IR) provided with the toolbox (see Table 1a). As this process introduces an IR-dependent delay, timestamps are adjusted using the IR's mean group delay.

High-pass Filter. Applies a linear-phase high-pass filter constructed using the Hamming window method. Parameters are stop and pass band edge frequencies. The output is cropped to achieve zero phase, hence no time-warp.

Low-pass Filter. Analogous to High-pass Filter.

MP3 Compression. Compresses the audio data to an MP3 file with a specified bit rate using the Lame encoder [3]. The encoder and decoder delays are compensated for by also using Lame as the decoder.

Saturation. Applies the transformation $x \leftarrow \sin(\pi x)$ multiple times (as specified) to each sample $x \in [-1, 1]$, which drives the sample towards the margins of -1 or 1, resulting in a simple imitation of a saturation effect.

[3] `http://lame.sourceforge.net`

```
function degr_cfg = demoDegradation()
  degr_cfg(1).methodname = 'degradationUnit_1';
  degr_cfg(1).parameter.someParam1 = 3;
  degr_cfg(2).methodname = 'degradationUnit_2';
  degr_cfg(2).parameter.someParam2 = 4;
```

a) Specifying a degradation.

```
audio_out = applyDegradation(
           'demoDegradation',audio,samplingFreq);
```

b) Applying a degradation to audio data.

Figure 2: Demo degradation: specification and application.

Speed-up (W). The signal is resampled at a specified sampling rate but returned using the original sampling rate, which results in a speed-up (or slow-down). Timestamps are adjusted accordingly.

Wow Resampling (W). Similar to Speed-up, but the resampling frequency is time-dependent: it oscillates around the original sampling rate at a specified frequency and amplitude, imitating non-constant speed in record players or tape machines. Timestamps are non-linearly warped to correspond to the output audio.

These degradation units implement audio and ground-truth transformation and thus form the building blocks for higher-level degradations, the subject of the next subsection.

2.2 Degradations

A degradation is a chain of degradation units with fixed parameters. The purpose of daisy-chaining degradation units is to allow the creation of more complex degradations than would be possible by using the degradation units alone. A degradation is defined in a Matlab function that acts as a configuration file describing the order and parameters of all degradation units used. An example is given in Figure 2a: the demo degradation specifies a first degradation unit `degradationUnit_1` with the parameter `someParam1` set to 3, and a second degradation unit `degradationUnit_2` with the parameter `someParam2` set to 4. In this way, the audio processing chain can be precisely specified. Any degradation thus defined can be applied to audio using the Matlab function `applyDegradation` provided by the ADT, see Figure 2b. Based on the degradation name, given as the first argument, the function retrieves the degradation definition in the struct array `degr_cfg` and cascades the degradation units in the specified order. Optionally, timestamp data can be supplied, which is also sequentially transformed to match the output audio, which is useful to make ground-truth usable on the degraded audio (see Section 2.1). The ADT comes with a range of pre-defined degradations. For the purpose of this paper we focus on the subset of six *'Real World Degradations'* that cover a variety of scenarios (more precise definitions come with the ADT source code).

Live Recording. Based on two degradation units: 1. Apply Impulse Response, using an IR of a large room ('Great Hall', taken from [20] and included in the ADT), 2. Add Noise: adding light pink noise.

Radio Broadcast. Based on two degradation units: 1. Dynamic Range Compression at a medium level to emulate the high loudness characteristic of many radio stations, 2. Speed-up, by 2%, which is commonly applied to music in commercial radio stations to shorten the music to create more advertisement time.

	correct	incorrect	not identified
Original	**100**	0	0
Live	0	0	100
Radio	3	3	94
PhonePlay	0	1	99
PhoneRec	5	7	88
MP3	**100**	0	0
Vinyl	4	0	96

Table 2: EchoNest audio ID results for 100 test songs.

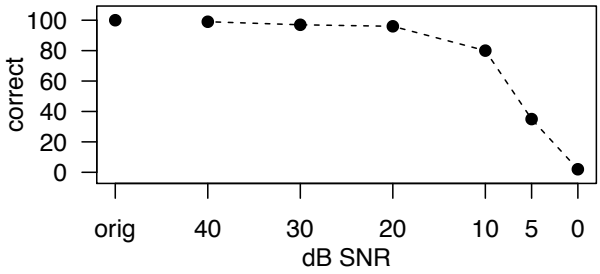

Figure 3: EchoNest audio ID results with pink noise.

Smartphone Playback. Based on two degradation units simulating a user playing back audio on a smartphone: 1. Apply Impulse Response, using the IR of a smartphone speaker ('Google Nexus One', Table 1a), which has a high-pass characteristic and a cutoff at ≈ 500Hz. 2. Add Noise, adding light pink noise.

Smartphone Recording. Based on four degradation units, simulating a user holding a phone in front of a speaker: 1. Apply Impulse Response, using the IR of a smartphone microphone ('Google Nexus One', Table 1a), 2. Dynamic Range Compression, to simulate the phone's auto-gain, 3. Clipping, 3% of samples, 4. Add Noise, adding medium pink noise.

Strong MP3 Compression. Based on one degradation unit: MP3 Compression at a constant bit rate of 64 kbps.

Vinyl. Based on four degradation units: 1. Apply Impulse Response, using a typical record player impulse response (Table 1a), 2. Add Sound, adding record player crackle (Table 1b), 3. Wow Resample, imitating wow-and-flutter, with the wow-frequency set to 33 rpm (speed of Long Play records), 4. Add Noise, adding light pink noise.

3. APPLICATIONS

In order to illustrate the insights that can be gained by using the ADT we evaluated several methods for standard music informatics tasks on suitable audio data. The results and brief discussions are given below.

3.1 Audio Identification Service

As a proof of concept we used the free web-based audio identification (audio ID) service from the Song API [4] pro-

[4] `http://developer.echonest.com/docs/v4/song.html`

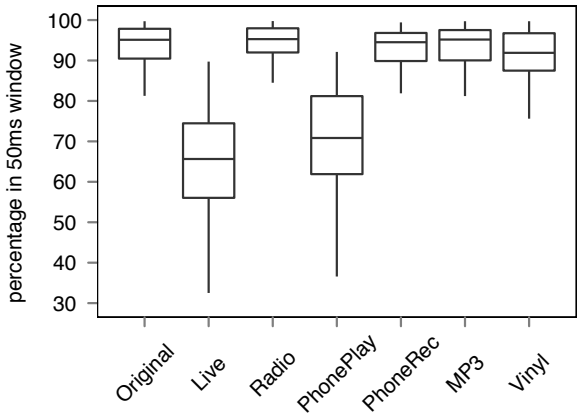

Figure 4: Score-to-Audio alignment accuracy under *Real World Degradations*. The boxes indicate the 1st, 2nd (median) and 3rd quartiles, the whiskers extend to 'the most extreme data point which is no more than 1.5 times the interquartile range' (R software [19]).

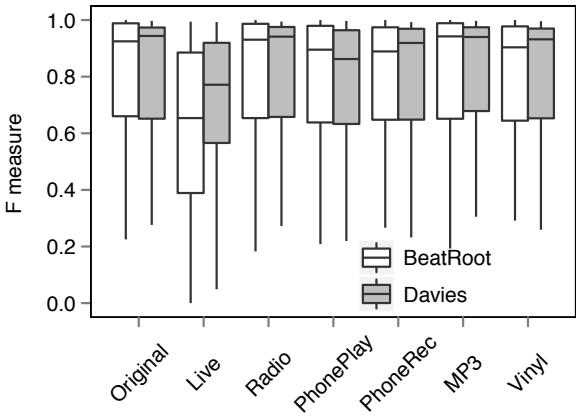

Figure 5: Comparison of beat-tracking performance under *Real World Degradations*.

vided by the company EchoNest. A user can compute an audio fingerprint using a dedicated program (ENMFP method) and query the EchoNest database for a corresponding identification number, artist and track name. If the system cannot identify a recording, that information is also returned.

We queried the database using 100 original rock and pop songs [5] taken from commercial CDs as well as degraded versions using the *Real World Degradations* defined in Section 2.2. The returned metadata was manually validated. The design goals for the EchoNest Audio ID service are clearly reflected in the results in Table 2: all 100 original tracks are correctly identified, as were all 100 files with strong MP3 compression. In contrast, all other degradations led to a clear failure with at most five recordings identified correctly, and low precision with up to seven recordings identified incorrectly. [6] However, as illustrated in Figure 3, an additional test showed that the system is reasonably robust against added pink noise up to a signal-to-noise ratio (SNR) of 10dB, for which 80 pieces were still correctly recognised (Figure 3). The poor real-world results are not surprising, since the service is supposed to *discriminate* between versions of the same song, not to detect similar songs. By contrast, the music informatics tasks below are concerned with the extraction of musical attributes that should persist even in degraded audio.

3.2 Score-to-Audio Alignment

Given a score and an audio recording for a piece of music, the aim of score-to-audio alignment is to find, for every position in the score, the corresponding position in the recording. In contrast to the audio ID task, which assumes a particular recording, the task is meant to work on any rendition or recording of the same musical work, and hence we expect a higher robustness against our real-world degradations. We use all 50 pieces from the Saarland Music Data [16], which contains audio recordings and corresponding MIDI files, both recorded using a Yamaha Disklavier. Our experimental setup is similar to the one described in [8]: the MIDI files are temporally distorted by randomly changing their tempo in 10-second intervals by up to 50%, faster or slower. We compute the alignment between the distorted MIDI files and the original audio recordings using the method described in [8], which combines chroma features with onset features. To measure the alignment accuracy, we computed for each recording the percentage of notes with an alignment error of less than 50ms for the onset position. The distribution of these values over all files is shown in the form of box-and-whisker plots in Figure 4.

This score-to-audio method is more robust than the EchoNest audio ID retrieval. The median over all files remains greater than 90% for all degradations, with two exceptions: *Live* and *Smartphone Playback*. Here, it is interesting to investigate the underlying reasons. The method employs a relatively simple onset detector to refine the alignment. The room IR used in the *Live* setting contains several early reflections, which generates several closely located 'copies' of onsets. These can easily be confused with the original onset. In the *Phone Playback* scenario, the significantly lower performance might be a result of applying the impulse response for the phone's speaker, which strongly attenuates all frequencies below 500 Hz including all fundamental frequencies up to B4. This leads to substantial differences between the observed audio and audio expected based on the score.

3.3 Beat-tracking

The aim of beat-tracking is to automatically find the timestamps of all beats in a piece of music. We compare two beat-trackers: *BeatRoot* [7] [5] and *Davies* [4] (QM Vamp Plugins implementation). *BeatRoot* first estimates note onset times, and forms a large number of tempo and beat hypotheses based on these onsets. A multiple-agent architecture is then used to determine the final beat estimates from the hypotheses. The *Davies* beat-tracker does not directly work on onsets but uses a continuous mid-level representation of onset salience, on which a comb filter is used to calculate the salience of different beat periods and beat alignments. Dynamic programming is used to retrieve

[5] A list of files is available on the project's website.

[6] In all cases, the songs are still easily recognisable to human listeners.

[7] http://www.eecs.qmul.ac.uk/~simond/beatroot, vers. 0.5.8

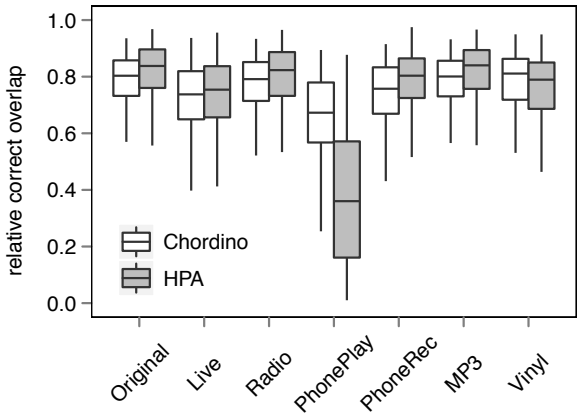

Figure 6: Chord detection performance: *Real World Degradations*

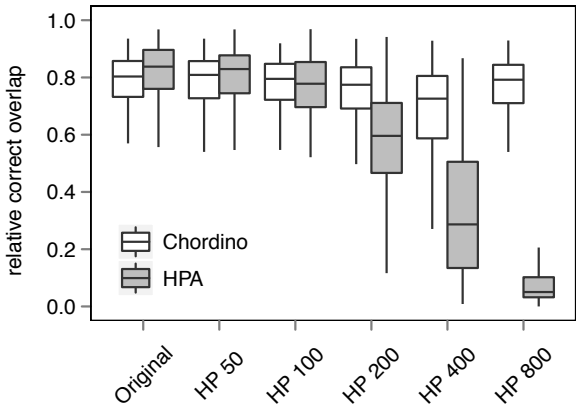

Figure 7: Chord detection performance: *High-pass filter* degradations.

the final beat estimate. Due to its dependence on onsets we would expect *BeatRoot* to be particularly susceptible to the *Live* setting (see Section 3.2).

We prepared 180 songs by the Beatles by degrading them using the *Real World Degradations*, resulting in 1260 wav files. Beats were extracted with both beat-trackers. We used a $\pm$70ms tolerance window to calculate the F measure for every song against human annotated ground-truth [14]. Figure 5 shows box-and-whisker plots of the F measure distributions, by degradation and beat-tracking method. For the original audio and most of the *Real World Degradations*, both beat-tracking methods show good performance, with median F measures always exceeding 0.85. With similar medians and inter-quartile ranges neither method has a clear advantage. The obvious exception is the *Live Recording* degradation, where the median F measure of both methods is substantially lower. *BeatRoot*: 0.65 (original: 0.92); *Davies*'s: 0.77 (original: 0.94). As explained in the case of score-to-audio alignment (Section 3.2), the likely cause are spurious onsets introduced by the impulse response; the *Davies* beat-tracker, which does not work on discrete onsets, is less affected.

3.4 Chord Detection

Chord detection is concerned with the transcription of the chord sequence in a piece of music. We test two different chord detection tools: *Chordino* [8] [15] and *HPA* [9] [17]. *Chordino* uses *NNLS Chroma* as a low-level feature, then matches manually defined chord templates to the chroma. Chords are modelled as hidden states in a hidden Markov model, and smoothing is achieved using Viterbi-decoding. *HPA* uses the same basic architecture, with some distinct differences: a beat-quantised, perceptually-inspired chroma representation (*HPA chroma*); a more complex probabilistic model that involves key and bass context; machine-learned chord profiles and transition parameters.

We continue to use the 180 songs by the Beatles from our beat-tracking experiment, as chord annotations are also available for them [10]. The chord detection outputs are evaluated by calculating the relative correct overlap for every song using a *MIREX-style* major/minor scheme [15].

Figure 6 shows box-and-whisker plots of the song-wise results by degradation and method. For the original audio and most degradations, *HPA* consistently outperforms *Chordino*, possibly due to its advanced exploitation of musical context and machine learning. Unlike the beat-trackers, both chord detection methods are relatively robust to the *Live Recording* degradation, with medians dropping less than 10 percentage points: *Chordino* 0.74 (original: 0.80), *HPA*: 0.75 (original: 0.84). Instead, they falter on the *Smartphone Playback* degradation: *Chordino* 0.67 (original: 0.80), *HPA*: 0.36 (original: 0.84). In order to understand whether this drop was caused by the degradation's high-pass characteristic (compare Section 3.2), we calculated five further degradations using the *High-pass Filter* degradation unit with the stop band edge parameter set to 50, 100, 200, 400 and 800 Hz, respectively. Figure 7 shows that the methods react very differently. The *Chordino* method remains relatively robust with the lowest median, 0.73, for a 400Hz stop band edge. The *HPA* method's advantage over *Chordino* is maintained for the 50Hz filter, but increasingly fails for higher cutoff frequencies with median values of 0.60 (200Hz), 0.29 (400Hz) and 0.05 (800Hz). In order to locate the reason for the strong drop-off, we studied the *HPA* chroma feature. Figure 8 shows an example of how the high-pass filter strongly affects the character of the feature, obfuscating the clear C major and A minor patterns.

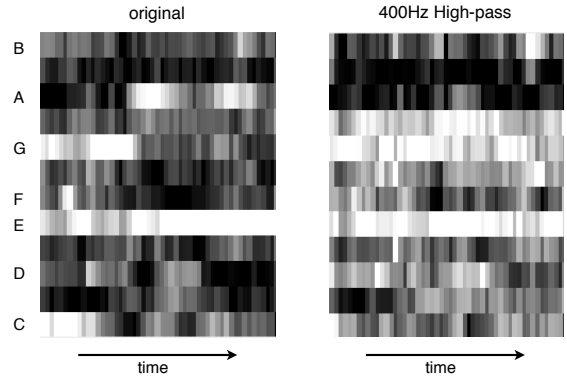

Figure 8: *HPA chroma* for the original and a high-pass filtered version of a snippet from 'Misery' by the Beatles.

[8] `http://isophonics.net/nnls-chroma`, Version 0.2.1

[9] `https://patterns.enm.bris.ac.uk/hpa-software-package`, Version 1.0

3.5 Summary of Experiments

As demonstrated by these examples, the ADT can be a valuable tool for investigating the robustness of music processing methods against a range of audio degradations. The *Real World Degradation* set in particular helps to provide an overview of the main problem areas for a method. For example, it has revealed that the otherwise excellent chord detection method *HPA* is not very robust to high-pass filtering (see Section 3.4). This gives an interesting example for how evaluating audio processing methods under multiple degradations can reveal their weaknesses and how it could drive the development of improved algorithms. While the ADT cannot replace 'looking at the data', experiments like the ones conducted in Section 3 can lead to interesting directions for further investigation.

4. CONCLUSIONS

In this paper, we have introduced and described the Audio Degradation Toolbox. The toolbox is written in Matlab and consists of 14 degradation units which can be chained into more complex degradations. Several predefined degradations can be used to simulate real-world degradations. With all required sounds, impulse responses and code modules included, the toolbox is entirely self-contained. Existing ground-truth can easily be transformed to account for time distortions introduced by the degradations.

We used the toolbox to analyse several methods in four music informatics tasks: audio ID, score-to-audio alignment, beat-tracking and chord detection. Our analysis offers insights into the robustness of different methods in various scenarios. In particular, we showed that a performance advantage of one method over another can be substantially reduced or even reversed by quasi-real-world degradations. We demonstrate how to track down particular weaknesses and argue that using the toolbox in this way makes it a powerful tool for developing more robust audio technology.

We are currently working on extending the toolbox with new sounds and degradation definitions. Furthermore, we plan to conduct user studies to investigate whether the degradations we propose have an influence on the way a human annotates an audio recording.

Acknowledgements: Matthias Mauch is funded by a Royal Academy of Engineering Research Fellowship, Sebastian Ewert is funded by EPSRC grant EP/J010375/1.

5. REFERENCES

[1] J. Burgoyne, J. Wild, and I. Fujinaga. An expert ground-truth set for audio chord recognition and music analysis. In *Proc. 12th Intl. Society for Music Information Retrieval Conf. (ISMIR)*, 2011.

[2] M. Casey, B. Fields, K. Jacobson, and M. Sandler. The effects of lossy audio encoding on genre classification tasks. In *Proc. 124th Convention of the Audio Engineering Society*, 2008.

[3] V. Chandrasekhar, M. Sharifi, and D. A. Ross. Survey and evaluation of audio fingerprinting schemes for mobile query-by-example applications. In *Proc. 12th Intl. Society for Music Information Retrieval Conf. (ISMIR)*, pages 801–806, 2011.

[4] M. Davies and M. Plumbley. Context-dependent beat tracking of musical audio. *IEEE Transactions On Audio Speech And Language Processing*, 15(3):1009–1020, 2007.

[5] S. Dixon. Evaluation of the audio beat tracking system BeatRoot. *Journal of New Music Research*, 36(1):39–50, 2007.

[6] C.-T. Do, D. Pastor, and A. Goalic. A novel framework for noise robust ASR using cochlear implant-like spectrally reduced speech. *Speech Communication*, 54(1):119–133, 2012.

[7] J. Downie, A. Ehmann, M. Bay, and M. Jones. The Music Information Retrieval Evaluation eXchange: Some observations and insights. In *Advances in Music IR*. 2010.

[8] S. Ewert, M. Müller, and P. Grosche. High resolution audio synchronization using chroma onset features. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1869–1872, 2009.

[9] D. Giannoulis, E. Benetos, D. Stowell, and M. D. Plumbley. IEEE AASP CASA challenge: Public dataset for scene classification task, 2012.

[10] C. Harte, M. Sandler, S. A. Abdallah, and E. Gómez. Symbolic representation of musical chords: A proposed syntax for text annotations. In *Proc. 6th Intl. Conf. on Music Information Retrieval (ISMIR)*, pages 66–71, 2005.

[11] H.-G. Hirsch and D. Pearce. The Aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions. In *ISCA Tutorial and Research Workshop (ITRW)*, 2000.

[12] K. Jacobson, M. Davies, and M. Sandler. The effects of lossy audio encoding on onset detection tasks. In *Proc. 125th Convention of the Audio Engineering Society*, 2008.

[13] P. v. Kranenburg, J. Garbers, A. Volk, F. Wiering, L. Grijp, and R. Veltkamp. Towards integration of music information retrieval and folk song research. Technical Report UU-CS-2007-016, Utrecht University, 2007.

[14] M. Mauch, C. Cannam, M. Davies, S. Dixon, C. Harte, S. Kolozali, D. Tidhar, and M. Sandler. OMRAS2 metadata project 2009. In *Late-breaking session, 10th Intl. Conf. on Music Information Retrieval (ISMIR)*, 2009.

[15] M. Mauch and S. Dixon. Approximate note transcription for the improved identification of difficult chords. In *Proc. 11th Intl. Society for Music Information Retrieval Conf. (ISMIR)*, pages 135–140, 2010.

[16] M. Müller, V. Konz, W. Bogler, and V. Arifi-Müller. Saarland music data (SMD). In *Late-breaking session, Intl. Society for Music Information Retrieval Conf. (ISMIR)*, 2011.

[17] Y. Ni, M. McVicar, R. Santos-Rodriguez, and T. De Bie. An end-to-end machine learning system for harmonic analysis of music. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(5):1771–1783, 2012.

[18] P. Proutskova and M. Casey. You call that singing? Ensemble classification for multicultural collections of music recordings. *Proc. 10th Intl. Conf. on Music Information Retrieval (ISMIR)*, pages 759–764, 2009.

[19] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, 2013.

[20] R. Stewart and M. Sandler. Database of omnidirectional and b-format room impulse responses. In *Proc. IEEE Intl. Conf. on Acoustics Speech and Signal Processing (ICASSP)*, pages 165–168, 2010.

[21] T. Wilmering, G. Fazekas, and M. Sandler. The effects of reverberation on onset detection tasks. In *Proc. 128th Convention of the Audio Engineering Society*, 2010.

[22] U. Zölzer, editor. *DAFx—Digital Audio Effects*. John Wiley & Sons, 2002.