# Scalable Perceptual Mixing and Filtering of Audio Signals using an Augmented Spectral Representation

Nicolas Tsingos

# SCALABLE PERCEPTUAL MIXING AND FILTERING OF AUDIO SIGNALS USING AN AUGMENTED SPECTRAL REPRESENTATION

*Nicolas Tsingos*

REVES - INRIA
Sophia Antipolis, France
nicolas.tsingos@sophia.inria.fr

**ABSTRACT**

Many interactive applications, such as video games, require processing a large number of sound signals in real-time. This paper proposes a novel perceptually-based and scalable approach for efficiently filtering and mixing a large number of audio signals. Key to its efficiency is a pre-computed Fourier frequency-domain representation augmented with additional descriptors. The descriptors can be used during the real-time processing to estimate which signals are not going to contribute to the final mixture. Besides, we also propose an importance sampling strategy allowing to tune the processing load relative to the quality of the output. We demonstrate our approach for a variety of applications including equalization and mixing, reverberation processing and spatialization. It can also be used to optimize audio data streaming or decompression. By reducing the number of operations and limiting bus traffic, our approach yields a 3 to 15-fold improvement in overall processing rate compared to brute-force techniques, with minimal degradation of the output.

## 1. INTRODUCTION

Many interactive applications such as video games, simulators and visualization/sonification interfaces require processing a large number of input sound signals in real-time (e.g., for spatialization). Typical processing includes sound equalization, filtering and mixing and is usually performed for each of the inputs individually. In modern video games, for instance, hundreds of audio samples and streams might have to be combined to re-create the various spatialized sound effects and background ambiance. This results in both a large number of arithmetic operations and heavy bus traffic. Although consumer-grade audio hardware can be used to accelerate some pre-defined effects, the limited number of simultaneous hardware voices calls for more sophisticated voice-management techniques. Besides, contrary to their modern graphics counterparts, consumer audio hardware accelerators still implement fixed-function pipelines which might eventually limit the creativity of audio designers and programmers. Hence, designing efficient software solutions is still of major interest.

While perceptual issues have been a key aspect in the field of audio compression (e.g., mp3) [1, 2], most software audio processing pipelines still use brute-force approaches which are completely independent of the signal content. As a result, the number of audio streams they can process is usually limited rapidly since the amount of processing cannot be adapted on-demand to satisfy a predefined time vs. quality tradeoff. This is especially true for multi-media or multi-modal applications where only a small fraction of the CPU-time can be devoted to audio processing.

In recent years, several contributions have been introduced that aim to bridge the gap between perceptual audio coding and audio processing in order to make audio signal processing pipelines more efficient. A family of approaches proposed to directly process perceptually-coded audio signals [3, 4, 5, 6, 7] yielding faster implementations than a full decode-process-recode cycle. Although they are well suited to distributed applications involving streaming over low-bandwith channels, they require specific coding of the filters and processing. Moreover, they cannot guarantee an efficient processing for a mixture of several signals, nor that they would produce an "optimal" processing for the mixture.

Others, inspired by psycho-acoustic research and audio coding work, tried to use perceptual knowledge to optimize various applications. For instance, a recent paper by Tsingos et al. [8] proposed a real-time voice management technique for 3D audio applications which evaluates audible sound sources at each frame of the simulation and groups them into clusters that can be directly mapped to hardware voices. Necessary sub-mixing of all sources in each cluster is done in software at fixed-cost. Dynamic auditory masking estimation has also been successfully used to accelerate modal synthesis [9, 10]. In the context of long FIR filtering for reverberation processing, the recent work by Lee et al. [11] also shows that significant improvement can be obtained by estimating whether the result of the convolution is below hearing threshold, hence reducing the processing cost. In this paper, we build on these approaches and propose a scalable, perceptually-based, audio processing strategy that can be applied to a frequency-domain processing pipeline performing filtering and mix-down operations on a large number of input audio signals [12]. Key to our approach is the choice of a signal representation that allows its progressive encoding and reconstruction. In this paper, we use Fourier-transform domain as a convenient and widely used solution which satisfies these constraints. In this context, we present a set of techniques to:

- dynamically maintain features of the input audio signals to process (for instance, based on pre-computed information on the input audio samples in a way similar to [8]),
- dynamically evaluate auditory masking between a number of input audio frames that have to be processed and mixed-down to produce a frame of audio output,
- implement a scalable processing pipeline by fitting a predefined budget of operations to the overall task based on the importance of each input audio signal.

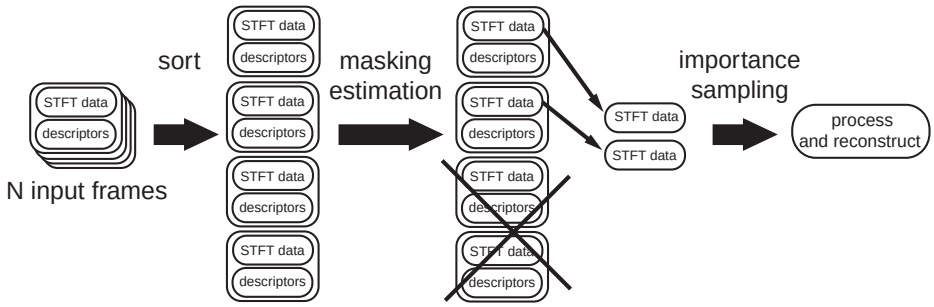


Figure 1: Overview of our scalable perceptual pipeline. All input signals frames at time $t$ are first sorted according to their energy content and a masking estimation is performed. Audible frames are then sampled according to an importance metric so that only a subpart of their pre-computed STFT coefficients are processed to produce the output frame at $t$.

## 2. OVERVIEW OF OUR APPROACH

Our approach can be decomposed into four main stages (see Figure 1). The first stage builds a frequency domain representation of the audio signals based on a short-time Fourier transform (STFT). This representation is augmented by a set audio descriptors such as the root-mean-square (RMS) level of the signal in several frequency bands and the tonality [1] of the signal. This kind of augmented description of audio signals is also similar in spirit to prior work in indexing and retrieval of audio [13]. This first stage is usually performed off-line when the signals to process are known in advance.

The three remaining stages: masking evaluation, importance sampling and actual processing are performed on-line during the interactive application. Audio signals are processed using small frames of audio data (typically using windows of 20 to 40 ms) and, as a consequence, all three later steps are performed for each frame of the computed output stream.

The masking step determines which subset of the input audio frames will be audible in the final mixture. It is not mandatory but usually makes the importance sampling step more efficient. It can also be used to limit bus traffic since all inaudible signals can be directly discarded after the masking evaluation and do not have to go through the actual processing pipeline.

The importance sampling step determines the amount of data to select and process in each input signal in order to fit the predefined operation budget and minimize audible degradations. Finally, the actual processing step performs a variety of operations on the audio data prior to the final mix-down.

In the remainder of the paper, sections 3 to 5 detail these four stages while section 6 presents example applications of our techniques in the context of equalization/mixing, reverberation processing and 3D audio rendering.

## 3. PRE-PROCESSING AUDIO SIGNALS

The first stage of our approach aims at pre-computing a signal representation from which the later real-time operations can be efficiently performed. We chose a representation based on a STFT of the input signals augmented with additional information.

### 3.1. Constructing the representation

For each frame of the input audio signal, we first compute the STFT of the audio data. For 44.1 kHz signals, we use 1024 sample Hanning-windowed frames with 50% overlap, resulting in 512 complex values in frequency domain. From the complex STFT, we then compute a number of additional descriptors:

- RMS level for a predefined set of $i$ frequency bands (e.g., octave or Bark bands),
- Tonality $T$ calculated as a spectral flatness measure [1]; tonality is a descriptor in $[0, 1]$ encoding the tonal (when close to 1) or noisy (when close to 0) nature of the signal,
- Reconstruction error indicator $Err$; this descriptor indicates how well the signal can be reconstructed from a small number of bins.

To compute the indicator $Err$, we first sort the FFT bins by decreasing modulus value. Then, several reconstructions (i.e., inverse Fourier transforms) are performed using an increasing number of FFT bins. The reconstruction error, calculated as the RMS level of the (time-domain) difference between the original and reconstructed frame, is then computed. For a $N$ bin FFT, we perform $k$ reconstructions using 1 to $N$ FFT bins, in $N/k$ increments. $Err$ is calculated as the average of the $k$ corresponding errors values. This indicator will be later used during the on-line importance sampling step.

Descriptors, together with the pre-sorted FFT bins, are computed for each frame of each input signal and pre-stored in a custom file-format. If required, descriptors can be stored separately from the FFT data used for the processing. They can be viewed as a compact representation of the signal, typically requiring a few additional kBytes of data per second of audio signals (e.g., 3kBytes/sec. at 44.1kHz for 1024 sample frames with 50% overlap and 8 frequency bands). Hence, for a set of short audio signals, they could easily fit into memory for fast random access over all signals.

### 3.2. Optimizing the representation

This representation can be further optimized if necessary during the pre-processing step. Frames whose energy is below audible threshold can be stored with a minimal amount of data. Basic masking calculations can also be performed while computing $Err$ by examining the signal-to-noise ratio between the energy in the selected FFT bins and the resulting reconstruction error. The number of stored bins can then be limited as soon as the signal-to-noise exceeds a specified threshold, which can further depend on the tonality of the signal [1]. Of course, any optimization made at this stage would imply that the signals are not going to be drastically

modified during the processing step. However, this restriction applies to any approach applied to audio data encoded using a lossy audio compression strategy.

Although compression is not the primary goal of our paper, we also experimented with various strategies to optimize storage space. By quantizing the complex FFT data with non-uniform 16-bit dynamic range and compressing all the data for each frame using standard compression techniques (e.g., zip), the size of the obtained sound files typically varies between 1.5 times (for wide-band sounds) and 0.25 times the size (e.g. speech) of the original 16-bit PCM audio data. If more dynamic range is necessary, it is also possible to quantize the $n$ first FFT bins, which contain most of the energy, over a 24-bit dynamic range and to represent the rest of the data using a more limited range with minimal impact on the size of the representation and quality of the reconstruction.

## 4. REAL-TIME MASKING EVALUATION

Once the input sound signals have been pre-processed, we can use the resulting information to optimize a real-time pipeline running during an interactive application.

The first step of our pipeline aims at evaluating which of the input signals are going to significantly contribute to a given frame of the output, which amounts to evaluating which input signals are going to be audible in the final mixture at a given time. Signals that have been identified as inaudible can be safely removed from the pipeline reducing both the arithmetic operations to perform and the bus traffic. Since the calculation must be carried on at each processing frame, it must be very efficient so that it does not result in significant overhead. The masking algorithm is similar to the one presented in [8] and makes use of the the pre-computed descriptors (see Section 3.1) for maximum efficiency.

First, all input frames are sorted according to some importance metric. In [8], a loudness metric was used but some of our recent experiments seem to indicate that the RMS level would perform equally well, if not better on average, for lack of a "ultimate" loudness metric [14]. If the signals must undergo filtering or equalization operations, we dynamically weight the RMS level values pre-computed for several frequency-bands to account for the influence of the filtering operations in each band. We can then compute the importance as the sum of all weighted RMS values.

Second, all signals are considered in decreasing importance order for addition to the final mixture according to the following pseudo-code:

$\mathbf{M_{mix}} = 200$
$\mathbf{P_{mix}} = 0$
$\mathbf{T} = 0$
$\mathbf{P_{toGo}} = \sum_k \mathbf{RMS_k}$
**while** $(dB(\mathbf{P_{toGo}}) > dB(\mathbf{P_{mix}}) - \mathbf{M}_{mix})$
**and** $(\mathbf{P_{toGo}} > \mathbf{ATH})$ **do**
tag signal $S_k$ as audible
$\mathbf{P_{toGo}} -= \mathbf{RMS_k}$
$\mathbf{P_{mix}} += \mathbf{RMS_k}$
$\mathbf{T} += \mathbf{P_k} * \mathbf{T_k}$
$\mathbf{T_{mix}} = \mathbf{T}/\mathbf{P_{mix}}$
$\mathbf{M_{mix}} = 27 * \mathbf{T_{mix}} + 6 * (1 - \mathbf{T_{mix}})$
k++
**end**

This process basically adds the level $RMS_k$ of each source to an estimate of the level of the final result in each band $P_{mix}$ (initially set to zero). Accordingly, it subtracts it from an estimate of the remaining level in each band $P_{toGo}$ (initially set to the sum of all RMS levels for all signals). The process stops when the estimated remaining level in each band is below a given threshold $M_{mix}$ from the estimated level of the final result. The process also stops if the remaining level is below the absolute threshold of hearing $ATH$ [15]. Threshold $M_{mix}$ is adjusted according to the estimated tonality of the final result $T_{mix}$, following rules similar to the ones used in perceptual audio coding [1]. In our applications, a simple constant threshold of -27 dB also gave satisfying results indicating that pre-computing and estimating tonality values is not mandatory. Note that all operations must be performed for each frequency band, although we simplified the given pseudo-code for the sake of clarity (accordingly, all quantities should be interpreted as vectors whose dimension is the number of used frequency bands and all arithmetic operations as vector arithmetic). In particular the process stops when the masking threshold is reached for *all* frequency bands.

## 5. IMPORTANCE SAMPLING AND PROCESSING

The second step of our pipeline aims at processing the sub-set of audible input signals in a scalable manner while preserving the perceived audio quality. In our case this is achieved by performing the required signal processing using a target number of operations over a limited sub-set of the original signal data. Note that it is not mandatory to perform masking calculations (as described in the previous section) in order to implement the following importance sampling scheme. However, the masking step limits the number of samples going through the rest of the pipeline and ensures that no samples will be wasted since our sampling strategy itself does not ensure that masked signals will receive a zero-sample budget.

### 5.1. Selecting a budget number of FFT bins

We can assume a constant number of arithmetic operations will be required for each complex FFT coefficient (i.e., bin). Hence, fitting a budget number of operations for our pipeline at each processing frame directly amounts to selecting a budget number of FFT bins for each frame of input sound signal. We can take advantage of pre-storing our FFT in decreasing energy order by directly processing the $n_i$ first FFT bins for each input signal $s_i$ so that $\sum_i n_i$ does not exceed our budget $N$.

We select $n_i$ as being directly proportional to an importance value $I_i$ calculated for each audio signal. In our case, we define $I_i$ as:

$$I_i = log(1 + E_i * (1 + Err_i)), \quad (1)$$

where $Err_i$ is the error indicator of signal $s_i$ and $E_i$ is the summed RMS level of the signal in all bands (including possible effects of filtering). As for the masking calculation, we use the RMS level value as the primary importance value. We further weight this value according to the error indicator of the signal, so that signals requiring more FFT bins to achieve a good reconstruction get higher priority.

The importance value $I_i$ is then normalized so that $\sum_i I_i = 1$ and the number of bins to process for each signal is simply obtained as: $n_i = N\ I_i$. Note that the overall target number is an upper bound and might not be exactly met. For instance, when an optimized input representation is used, some frames might contain

a number of FFT bins smaller than their calculated $n_i$. Currently, we do not re-affect the additional number of bins to another signal.

### 5.2. Processing and reconstruction

Once the most-important FFT bins have been selected for all signals according to our target budget, they are simply sent to the actual processing pipeline. All calculations are done in frequency-domain so that a single inverse FFT is required to obtain the final time-domain audio signal. Since we pre-compute FFT data for 50% overlapped frames, reconstructing a time-domain frame for playback requires processing two frequency-domain frames, involving two inverse FFTs and an overlap-add operation.

## 6. APPLICATIONS AND RESULTS

We implemented and tested our scalable processing algorithm for three applications: a simple mixing and equalization pipeline, an FIR reverberation pipeline and a massive spatial audio application which can render hundreds of simultaneous sound sources in real-time. Example results are available for listening at the following URL: www-sop.inria.fr/reves/projects/dafx05. All examples were implemented in *C++* without any specific optimization and tested on a standard laptop computer with a *Pentium 4m* 1.8 GHz processor. All processing was done using 32-bit floating point arithmetic. Sampling rate was 44.1 kHz and we used output frames of 1024 samples. Hence, the output rate for audio data was 43 Hz. Masking calculations were performed using 15 frequency subbands.

### 6.1. Mixing and equalization

Our first application performs simple equalization and mix-down operations on a number of input sound signals. In this case, all data was streamed and decompressed from the disk in real-time. Even for a relatively small number of signals to process (in our test example we used 8), our approach shows a 3-fold speed-up compared to processing the entire data set. Overall processing speed, including streaming from disk and final time-domain reconstruction, is doubled. Table 1 shows a compute-time breakdown for various stages of the pipeline when processing the STFT data at several "resolutions", decreasing the number of target FFT bins. The results, expressed in Hz, correspond to the rate at which a full frame of output can be calculated. The load rate corresponds to the rate at which the audio data can be streamed from disk. For the load rate and processing rate, the figures are given per input signal. The total rate is given for the entire process applied to all signals, including streaming from disk. In this case, the total rate is streaming-bound. Hence, using an optimized representation, as described in section 3.2, brings a more than 2-fold improvement (as can be seen in the last column). In this case, near-transparent processing could be achieved in 1/10th of the duration of an output frame.

| target FFT bins | all (4096) | 2000 | 500 | 500 (optim.) |
|---|---|---|---|---|
| load rate (Hz) | 1900 | 1896 | 1920 | 4500 |
| processing rate (kHz) | 60 | 116 | 180 | 200 |
| total rate (Hz) | 207 | 209 | 213 | 440 |
| avg. masked frames (%) | 0 | 15 | 15 | 15 |

Table 1: Breakdown performance figures for our test mixing and equalization pipeline. Load and processing rate are given per signal. All values are time-averaged over 29 seconds of processing-time.

### 6.2. Block FIR filtering and reverberation

Our second application targets long FIR filters as used for reverberation processing. A common technique to implement low latency convolution with long filters is to decompose the filter in successive smaller blocks. These blocks can be of constant length [16, 17] or can be adapted to optimize the number of arithmetic operations [18]. Our technique extends the recent approach by Lee et al. [11] by 1) determining which part of the reverberation filter will not be audible due to auditory masking and 2) allowing for scalable rendering of the reverberation. In fact, this application is very similar to our previous example: each block of the reverberation filter can be convolved with separate delayed copies of the original signal (by multiples of one frame) and the results are mixed together to produce one frame of reverberant output. We tested our perceptual reverberation algorithm with artificial reverberation FIR filters created from exponentially decaying white noise [19]. However, our approach could be applied to any measured impulse response. Table 2 shows results of our approach applied to a stereo rendering of up to 12-second long reverberation filters (corresponding to about 1000 blocks of 512 time-samples). In the corresponding examples, the exponential decay was chosen to obtain a reverberation time of about 4.5 seconds. Here again, we show a compute-time breakdown for various stages of the pipeline when processing the STFT data at several "resolutions", decreasing the number of target FFT bins. The two left-most columns, denoted "all" and "all (mask)" correspond respectively to full processing (i.e., reference) and processing of all FFT bins of all *audible* audio frames (i.e., masking is turned on but all audible data is processed).

| Rotor example | | | | | |
|---|---|---|---|---|---|
| target FFT bins | all | all (mask) | 50000 | 10000 | 5000 |
| masking/sampling rate (Hz) | 4754 | 2322 | 2263 | 2340 | 2387 |
| processing rate (Hz) | 82 | 263 | 809 | 1965 | 3108 |
| FFT rate (Hz) | 2109 | 2068 | 2124 | 2120 | 2110 |
| total rate (Hz) | 38 | 105 | 230 | 350 | 405 |
| avg. masked frames (%) | 0 | 87 | 87 | 87 | 87 |
| **Song example** | | | | | |
| target FFT bins | all | all (mask) | 50000 | 10000 | 5000 |
| masking/sampling rate (Hz) | 2452 | 1282 | 1273 | 1306 | 1316 |
| processing rate (Hz) | 40 | 165 | 570 | 1666 | 2556 |
| FFT rate (Hz) | 2126 | 2107 | 2084 | 2144 | 2129 |
| total rate (Hz) | 19 | 68 | 164 | 270 | 304 |
| avg. masked frames (%) | 0 | 80 | 80 | 80 | 80 |
| **Voice example** | | | | | |
| target FFT bins | all | all (mask) | 50000 | 10000 | 5000 |
| masking/sampling rate (Hz) | 13632 | 6387 | 6172 | 6004 | 6670 |
| processing rate (Hz) | 269 | 531 | 1115 | 2081 | 3881 |
| FFT rate (Hz) | 2130 | 2137 | 2162 | 2080 | 2200 |
| total rate (Hz) | 116 | 197 | 324 | 435 | 567 |
| avg. masked frames (%) | 0 | 93 | 93 | 93 | 93 |

Table 2: Breakdown performance figures for our test block FIR reverberation pipeline. All values are time-averaged over a processing-time equal to the duration of the input data.

As can be seen from the results, our approach can bring FIR-based reverberation to the efficiency level of IIR-based techniques [19] with an equivalent cost of about a few tens of operations/time sample. Our masking strategy also appears to be more efficient

than the simpler hearing-threshold cut-off presented in [11] with a reverberation cut-down of up to 93%. Although the degradation becomes noticeable at low processing rates, we found the overall perceived room-effect to be well preserved. We believe an optimized implementation of our approach, including specific assembly language processing and FFT code (we used a simple implementation from [20]) could outperform efficient convolution engines such as [21] for applications where lossy processing is acceptable.

### 6.3. Massive spatial audio rendering

We finally applied our technique to spatial audio rendering. In this case, we perform a complex multiply on our data to account for various filtering effects (e.g. Head Related Transfer Function [22], atmospheric scattering, occlusions, source directivity) and to delay the audio data according to the distance and direction of each incoming source signal. As in the reverberation application, our pipeline computes a binaural output and thus requires four inverse FFTs to reconstruct a full output frame of stereo audio data. Table 3 shows results of our approach when used to process 1000 3D sound sources, using a variable number of coefficients for the entire pipeline. In our test case, all sound sources are instances of 8 primary audio signals, demonstrating possible application to auralizing sound reflection or diffraction resulting from a geometrical acoustics simulation [23, 24]. Refresh rates for a simultaneous basic 3D-graphics rendering of the sound sources are also provided. As we reduce the number of audio operations, more CPU-time can be devoted to graphics rendering resulting in increased frame-rates. For this application, our approach can lead to 8 to 15 fold-improvement over the brute force techniques while maintaining good output quality. In particular, we estimate that our unoptimized approach can render up to 6 times as many 3D sound sources as the approach of [8] which relied on SSE-optimized code and hardware spatialization. Besides, it does not require specific spatial clustering. Example movie files demonstrating these results are also available on-line.

| target FFT bins | all (512000) | 80000 | 50000 | 10000 |
|---|---|---|---|---|
| audio rate (Hz) | 8 | 56 | 70 | 150 |
| graphics rate (Hz) | n.a. | 8 | 15 | 27 |
| avg. masked frames (%) | 0 | 24 | 24 | 24 |

Table 3: Performance figures for our test 3D audio-visual rendering application processing 1000 sound sources. Masking is turned-on for all test-cases except the reference (i.e., first column). Results are averaged over 11 sec. of processing-time.

## 7. DISCUSSION

One limitation of our importance sampling scheme is that it requires a fine-grain scalable model of sounds to be applicable. However, we also experimented with coarser-grain time-domain representations with promising results [14]. As with all frequency-domain processing approaches, it might require many inverse FFTs per frame to reconstruct multiple channels of output. This might be a limiting factor for applications requiring multi-channel output (e.g., 5.1 surround). However, in most cases the number of output channels is small.

Another limiting factor is that we use pre-computed information to limit the overhead of our frequency domain processing and final reconstruction. In the case were the input signals are not known in advance (e.g., voice over IP, real-time synthesis,...), an equivalent representation would have to be constructed on-the-fly prior to processing. We believe that if a small number of such streams are present, our approach would still improve the overall performance.

Pre-sorting the FFT data also implies that the processing should not drastically affect the frequency spectrum of the input signals which might not be the case. However, any approach using signals encoded with a lossy algorithm would suffer from the same limitation since perceptual (e.g., masking) effects would be encoded *a priori*. A solution to the problem would be to store sorted FFT data associated to a number of subbands and re-order them in real-time according to how filtering operations might affect the level in each subband. Sorting the output frequency-domain data would also be necessary if several effects have to be chained together. Another solution to better account for filtering effects would be to extend our importance sampling strategy to account explicitly for frequency content (currently, importance is implied by the pre-computed ordering of STFT data so that only the number of processed bins has to be determined). This might yield to an approach closer in spirit to [5] albeit we would still benefit from scalability and masking estimation (and would not require specific filter representation).

Although it can be optimized, our STFT data is not a compact representation (at least not as compact as standard perceptually coded representations, such as mp3 or AAC) which could be a limitation for streaming or bus transfers. However, masking calculations limit this problem, especially since they only require the additional descriptors to evaluate audible data.

Finally, the importance model used for selecting which FFT coefficients to process could be improved by using a more sophisticated loudness-related metric.

## 8. CONCLUSIONS

We presented a scalable approach to efficiently filter and mix-down a large number of audio signals in real-time. By pre-computing a Fourier frequency-domain representation of our input audio data augmented with a set of audio descriptors, we are able to concentrate our processing efforts on the most important components of the signals. In particular, we show that we can identify signals which will not be audible in the output at each processing time-frame. Such signals can be discarded thus reducing computational load and bus traffic. Remaining audible signals are sampled based on an importance metric so that only a subset of their representation is processed to produce a frame of output. Our approach yields a 3 to 15-fold improvement in overall processing rate compared to brute-force techniques with minimal degradation of the output. As future extensions, we plan to conduct perceptual validation studies to assess the auditory transparency of our approach at several "processing bit-rates" and further improve on our masking calculation and importance sampling metrics.

## 9. ACKNOWLEDGMENTS

The author would like to thank the anonymous reviewers for their useful comments. This research was supported in part by the 2-year RNTL project OPERA, co-funded by the French Ministry of Research and Ministry of Industry.
http://www-sop.inria.fr/reves/OPERA.

## 10. REFERENCES


[1] E. M. Painter and A. S. Spanias, "Perceptual coding of digital audio," *Proceedings of the IEEE*, vol. 88, no. 4, Apr. 2000.

[2] Jürgen Herre, "Audio coding - an all-round entertainment technology," in *Audio Engineering Society 22nd International Conference on Virtual, Synthetic and Entertainment Audio (AES'22), Espoo, Finland*, June 15-17 2002, pp. 139–148.

[3] C. A. Lanciani and R. W. Schafer, "Psychoacoustically-based processing of MPEG-I layer 1-2 encoded signals," in *Proc. IEEE Signal Processing Society 1997 Workshop on Multimedia Signal Processing*, June 1997, pp. 53–58.

[4] Chris A. Lanciani and Ronald W. Schafer, "Subband-domain filtering of MPEG audio signals," in *Proceedings of Intl. Conf. on Acoustics, Speech and Signal Processing*, Mar. 1999, pp. 917–920.

[5] Abdellatif B. Touimi, "A generic framework for filtering in subband domain," in *In Proceeding of IEEE 9th Workshop on Digital Signal Processing, Hunt, Texas, USA*, October 2000.

[6] Abdellatif B. Touimi, Marc Emerit, and Jean-Marie Pernaux, "Efficient method for multiple compressed audio streams spatialization," in *In Proceeding of ACM 3rd Intl. Conf. on Mobile and Ubiquitous multimedia*, 2004.

[7] D. Darlington, L. Daudet, and M. Sandler, "Digital audio effects in the wavelet domain," in *Proceedings of COST-G6 Conference on Digital Audio Effects, DAFX2002, Hamburg, Germany*, Sept. 2002.

[8] Nicolas Tsingos, Emmanuel Gallo, and George Drettakis, "Perceptual audio rendering of complex virtual environments," *ACM Transactions on Graphics (Proceedings of SIGGRAPH'04)*, vol. 23, no. 3, 2004.

[9] Mathieu Lagrange and Sylvain Marchand, "Real-time additive synthesis of sound by taking advantage of psychoacoustics," in *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-01), Limerick, Ireland, December 6-8*, 2001.

[10] Kees van den Doel, Dave Knott, and Dinesh K. Pai, "Interactive simulation of complex audio-visual scenes," *Presence: Teleoperators and Virtual Environments*, vol. 13, no. 1, 2004.

[11] Wen-Chieh Lee, Chung-Han Yang, Chi-Min Liu, and Juin-In Guo, "Perceptual convolution for reverberation," in *In Proceeding of 115th AES Convention, Los Angeles*, Oct. 2003.

[12] Udo Zölzer, Ed., *DAFX - Digital Audio Effects*, Wiley, 2002.

[13] Perfecto Herrera, Xavier Serra, and Geoffroy Peeters, "Audio descriptors and descriptors schemes in the context of MPEG-7," in *Procedings of International Computer Music Conference (ICMC99)*, 1999.

[14] Emmanuel Gallo, Guillaume Lemaitre, and Nicolas Tsingos, "Prioritizing signals for selective real-time audio processing," in *proceedings of Intl. Conf. on Auditory Display (ICAD) 2005, Limerick, Ireland*, July 2005.

[15] E. Zwicker and H. Fastl, *Psychoacoustics: Facts and Models*, Springer, 1999, Second Upadated Edition.

[16] Barry D. Kulp, "Digital equalization using fourier transform techniques," in *In Proceeding of 85th AES Convention, Los Angeles*, Nov. 1988.

[17] Jia-Sen Soo and Khee K. Pang, "Multidelay block frequency domain adaptive filter," *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 38, no. 2, Feb. 1990.

[18] G. García, "Optimal filter partition for efficient convolution with short input/output delay," in *In Proceeding of 113th AES Convention, Los Angeles*, Oct. 2002.

[19] Mark Kahrs and Karlheinz Brandenburg, Eds., *Applications of Digital Signal Processing to Audio and Acoustics*, Kluwer Academic Publishers, 1998.

[20] William Press, Saul Teukolsky, William Vetterling, and Brian Flannery, *Numerical Recipes in C, 2nd edition*, Cambridge University Press, New York, USA, 1992.

[21] Anders Torger, "BruteFIR software convolution engine," http://www.ludd.luth.se/ torger/brutefir.html.

[22] Durand R. Begault, *3D Sound for Virtual Reality and Multimedia*, Academic Press Professional, 1994.

[23] J.B. Allen and D.A. Berkley, "Image method for efficiently simulating small room acoustics," *J. of the Acoustical Society of America*, vol. 65, no. 4, 1979.

[24] N. Tsingos, T. Funkhouser, A. Ngan, and I. Carlbom, "Modeling acoustics in virtual environments using the uniform theory of diffraction," *ACM Computer Graphics, SIGGRAPH'01 Proceedings*, pp. 545–552, Aug. 2001.