March 15, 2017

Efficient externalized audio reverberation with smooth transitioning

Ian Kelly

Marcin Gorzel

Alper Güngörmüsler

Efficient externalized audio reverberation with smooth transitioning

ABSTRACT

Audio for immersive or simulated environments such as virtual reality (VR) matches the acoustics of the environment being simulated. Reverberative effects of a simulated environment are reproduced in a manner consistent with a corresponding real-world environment. For example, in a VR game, a user moves between a cathedral and a small room which are virtual environments that have substantially different acoustics. Providing audio reverb that matches a current virtual environment and smoothly transitions between virtual environments is essential for a quality VR experience. Providing smooth audio requires implementing a change in reverberation settings, e.g., filters, synchronously with gameplay which imposes a large computational load. This disclosure describes techniques to provide realistic audio reverberation with smooth transitions between simulated environments and low computational load. The techniques present different reverberation at each ear and position sound sources external to the user.

KEYWORDS

spatial audio; reverberation; virtual reality; convolutional reverb

BACKGROUND

Reverberation is the persistence of a sound after the sound is produced, caused by sound reflecting off several objects, which causes at first a buildup of intensity followed by a decay. The time taken for a sound signal to decay by 60 dB (a factor of one thousandth in sound pressure level) is symbolized as RT60. For a realistic rendering of simulated environments, electronic sound-generating devices introduce reverberation into the audio feed.

Reverberation is introduced by passing an audio feed through a digital filter known variously as “room impulse response,” or simply “impulse response,” or “reverb tail.” The reverb

tail matches the reverberative properties (e.g., sound-reflective, sound-absorptive, frequency-specific, etc.) of the environment being simulated. For example, a large cathedral has certain reverberative properties due to physical structure, materials used in construction, etc. Such properties are captured in a reverb tail that upon convolution with an incoming audio signal produces the simulated audio output. Such digital simulation of sound as heard in different spaces is important in VR applications to create a realistic, immersive user experience.

Generation of the reverb tail is computationally expensive. If a user switches the virtual environment to certain types of spaces such as large rooms with long echoes, it may be infeasible with limited computational power, such as on a mobile device, to generate reverb tails without delays that affect the quality of experience. An abrupt switch of reverb tails from one virtual space to another can generate a jarringly audible glitch.

DESCRIPTION

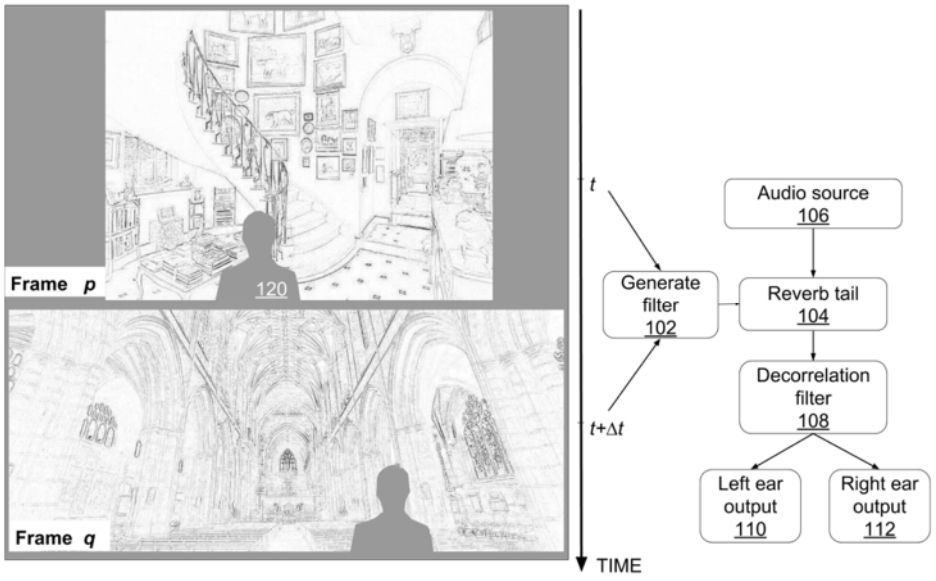

Fig. 1: Progression of a virtual reality sequence, with accompanying changes to audio reverberation filters

Fig. 1 depicts an example progression of a virtual reality sequence of video frames, with corresponding changes to audio reverberation filters. At a frame *p*, an avatar (120) is situated in a small room. At a later frame *q*, which can for example be an immediate next frame from frame *p*,

the avatar has transitioned to the nave of a large cathedral. Throughout the sequence of frames, an audio source (106) is active, delivering an accompanying soundtrack. In order to simulate acoustics of the environment depicted in the current frame, the audio source is convolved with a reverb tail (104).

The reverb tail is a digital representation of the acoustics that matches the characteristics of the environment depicted in the current frame. For example, if the environment is based on or similar to a real-world environment (e.g., a concert hall, a forest, etc.), the reverb tail represents the acoustics of such an environment. Further, if the environment is an entirely simulated environment (e.g., a mountain made of cotton), the reverb tail represents acoustics of such a simulated environment. The acoustics of frame *p* (small room) are very different from that of frame *q* (large cathedral), for example, a small room has hardly any echoes, whereas a large cathedral has very long and persistent echoes. The reverb tail is switched from small-room acoustics to large-cathedral acoustics by generating and applying as reverb tail a filter appropriate to the current VR frame (102).

To provide a high quality VR experience, the switch of reverb filters needs to be synchronous with the change in the visual frame sequence. Fast switching of filters is computationally difficult. If done abruptly, the switch in filters can result in a jarring audible glitch. Furthermore, the reverberations in the sound signals presented to the two ears need to be different in order to provide realistic, externalized audio. For this purpose, the output of the reverb tail is passed through a decorrelation filter (108), which produces decorrelated reverberations suitable for left ear (110) and right ear (112) output, e.g., playback through headphones.

The creation of a reverb tail, which commences upon a user (an avatar) entering a new VR space, is implemented as follows:

1. *N* vectors of wide-band Gaussian noise are created. The vectors are filtered, for example, using a biquad bandpass filter. After the filtering, each of the *N* vectors is a band-limited Gaussian, e.g., limited to one (different) octave each.
2. Each of the *N* band-limited vectors is scaled, sample-by-sample, by one of *N* exponential decay curves. The exponential decay curves each go down to a limit (e.g., the RT60 limit) at possibly differing rates of decay, and are based on acoustic properties of the environment being simulated. After the scaling, the *N* vectors are each a band-limited Gaussian, decaying in time at differing rates. The *N* vectors are summed to create a time-domain filter that decays at differing rates in different frequency bands. This time-domain filter, when convolved with the incoming audio, generates reverberations, and is capable of arbitrary RT60 in each frequency band.
3. To implement efficient real-time filtering of an incoming audio signal, the *N* band-limited, time-decaying Gaussians are converted to the frequency domain and stored as kernels of a pair of partitioned fast Fourier transform (FFT) filters.

The frequency-domain filters as described above are reverb tails that are convolved with the incoming audio signal to generate reverberations (convolutional reverb).

This disclosure describes low computational complexity techniques that perform smooth, glitch-free and real-time switch of reverb tails. To reduce computational load, the *N* Gaussian noise vectors are precomputed upon initialization, stored, and used whenever there is a switch in reverb tails. To eliminate glitches and reduce computational load during transition of reverb tails, the reverb tail is computed in chunks of length equal to the audio buffer of an implementing system such as a VR capable mobile phone or headset. Transition is carried out by updating, partition by partition, the kernels of the FFT filters over several compute cycles.

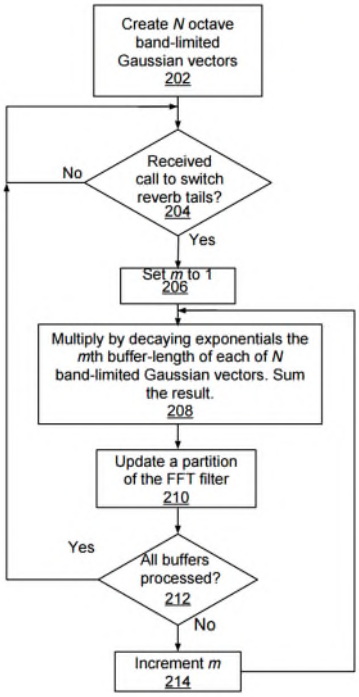

Fig. 2: Generating and updating reverb tails

Fig. 2 illustrates a process by which reverb tails are generated and switched. Upon initialization or system start-up (202), *N* Gaussian noise vectors are created and band-limited to an octave. Octave band-limitation of the *N* noise vectors can be performed, for example, by biquad bandpass filters that are each one octave wide. These pre-computed band-limited Gaussian noise vectors can be of any length as reverb tails can be produced from truncated or concatenated versions as needed. The lengths of the Gaussian noise vectors are chosen to be long enough to remove any periodicity that arise upon transformation to frequency domain. For example, these may be one-third the length of the longest expected reverb tail. Once created, the Gaussian noise vectors are converted to the frequency domain and stored for subsequent use. The Gaussian noise vectors may be represented at a suitable sampling rate, for example, 48 kHz, which, for a reverb tail duration of one second results in a storage requirement of 4.5 megabytes.

Upon receipt of a call to switch reverb tails (204), a process starts that switches, over the course of successive compute cycles, the entire reverb tail. At first, an index *m* for the buffer-

number is initialized (206). At each compute cycle, a length of reverb tail equal to one audio buffer is replaced as follows:

- The *m*th buffer-lengths of each of the *N* Gaussian noise vectors are multiplied by exponentials that decay down to the RT60 level, and the resulting vectors are summed element-by-element (208).
- The resultant sum-vector is used to update a partition of the FFT filter (210).

Since in a compute cycle only a portion of the FFT filter is updated, the FFT filter, over the period of transition, holds parts of both the old reverb tail as well as the new reverb tail. This gradual transition from old to new eliminates the glitch that is heard under single-cycle switch. Even as the transition from old to new reverb filter is under way, the audio feed continues to be processed, and hence reverberated, by the FFT filter. In some cases, the FFT filter and the new reverb tail differ in length. In such a case, the process described in Figs. 3-5 is followed during update of the FFT filter.

Once a partition of the FFT filter is updated, a test (212) is carried out to see if all audio buffers are processed. If yes, the system awaits receipt of a next call to switch the reverb tails. If not, the buffer-index *m* is updated (214) and steps 208-212 are repeated in the next compute cycle to update the next partition of the FFT filter.

Figs. 3-5 illustrate the transitioning process for different lengths of the new reverb tail and the frequency domain kernel. Fig. 3 depicts the case when length of the new reverb tail equals length of the frequency domain kernel; Fig. 4 depicts the case when the new reverb tail is longer than the frequency domain kernel; and Fig. 5 depicts the case when the new reverb tail is shorter than the frequency domain kernel. In each of Figs. 3-5, the circled-star symbol (*) represents convolution.

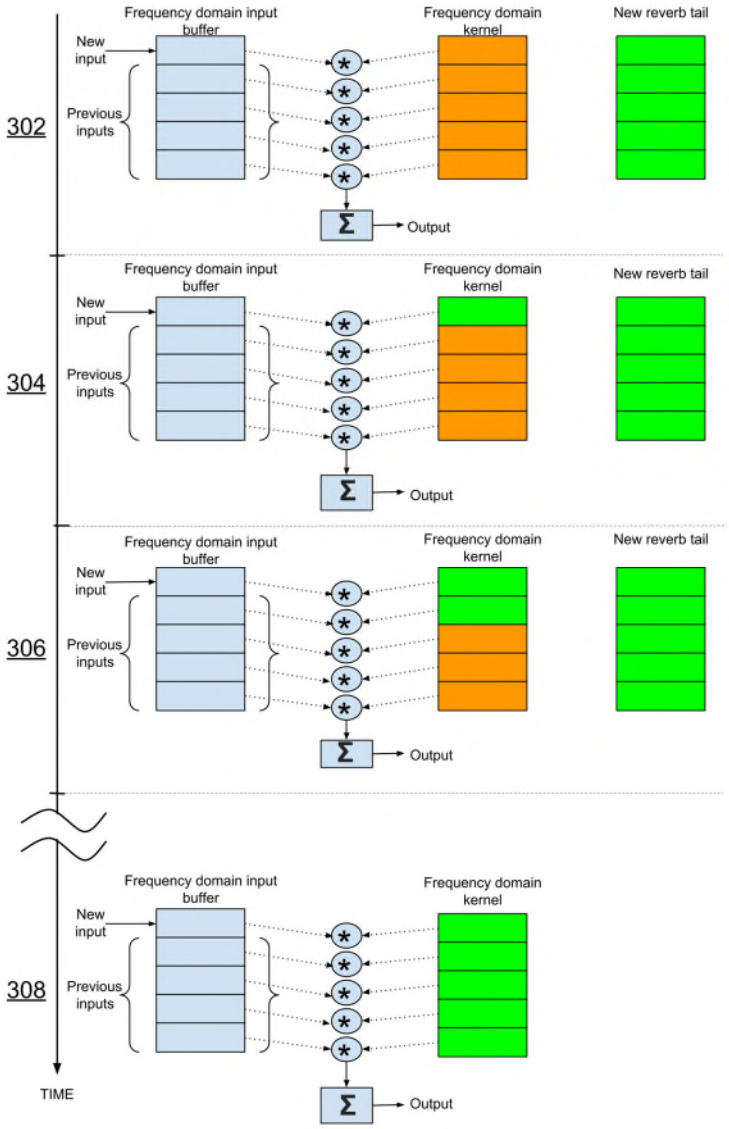

Fig. 3: Transition of new reverb tail into frequency domain kernel when lengths are equal

When the length of the new reverb tail equals that of the frequency domain filter, as in Fig. 3, the transition procedure is straightforward. At a certain compute cycle, (302) the old reverb tail is active, such that when convolved with contents of the frequency domain input buffer and result summed, a reverberation corresponding to the old reverb tail is generated. At the next compute cycle (304), a partition of the frequency domain kernel is populated with a buffer-length of the new reverb tail. The input buffer is updated, so that the top of the input buffer contains the latest audio input, while lower partitions of the input buffer contain past inputs. The convolution-and-sum

operation performed in compute cycle 304 results in a reverberation that is a combination of old and new reverb tails. At each subsequent compute cycle (306), a next partition of the frequency domain kernel is populated with the next buffer-length of the new reverb tail, the audio input buffer updated, and the convolution-and-sum operation executed. After several compute cycles, the old reverb tail is completely flushed out, and the new reverb tail transitioned in (308).

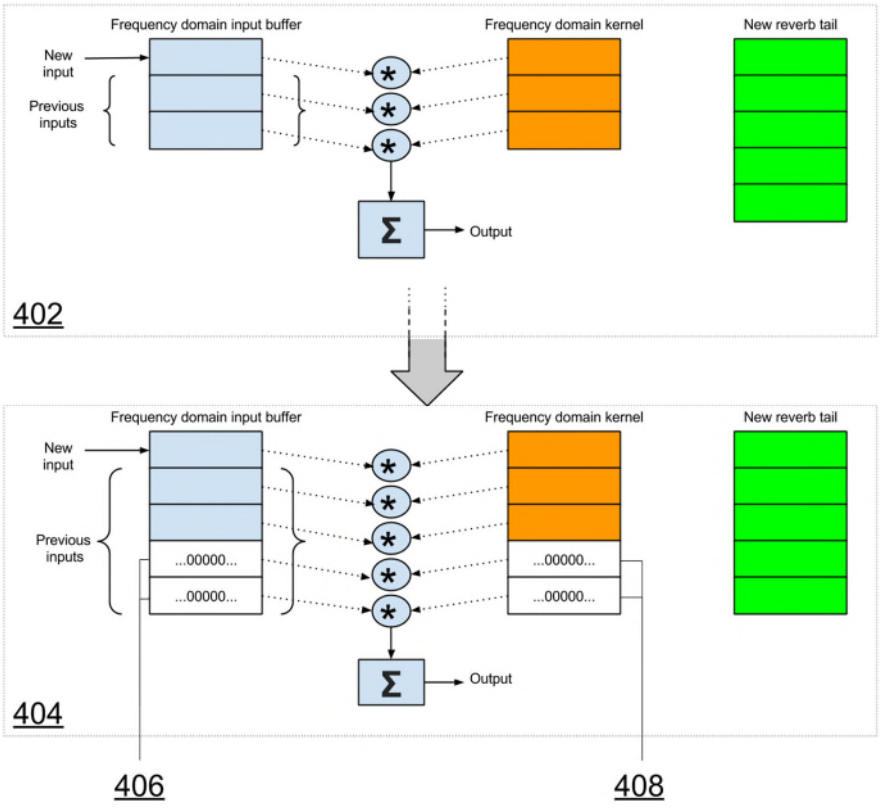

Fig. 4: Transition of new reverb tail into frequency domain kernel when the new reverb tail is longer in length than the frequency domain kernel

Fig. 4 illustrates a situation when the length of the new reverb tail is longer than that of the frequency domain filter. In such a case, both the input buffer (406) and the frequency domain kernel (408) are extended, e.g., by allocation of additional memories to match the length of the new reverb tail. The extended portions of the input buffer and the frequency domain kernel are populated with zeros. In this way, a mismatch in length (402) between the frequency domain filter and the new reverb tail is converted to matching lengths (404) for which the process of Fig. 3 applies.

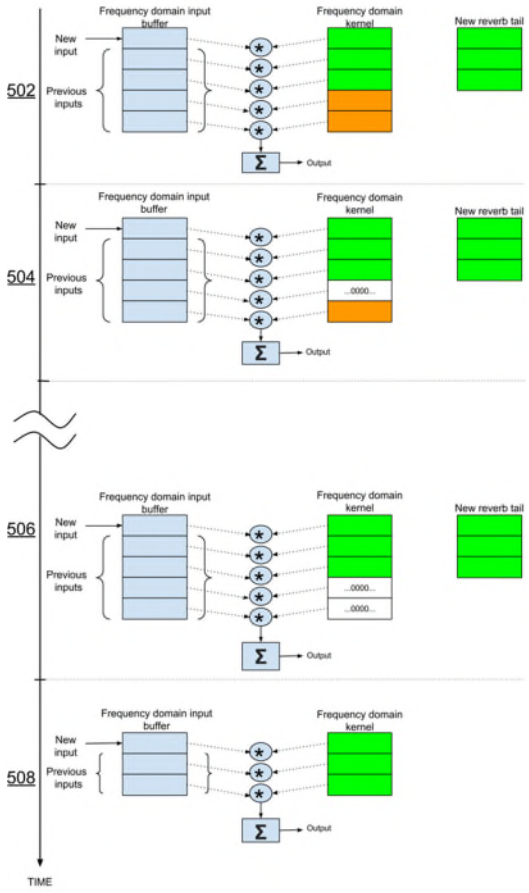

Fig. 5: The transition of the new reverb tail into the frequency domain kernel when the new reverb tail is shorter in length than the frequency domain kernel

Fig. 5 illustrates a situation when the length of the new reverb tail is shorter than that of the frequency domain filter. In this case, the transition of new reverb tail into the frequency domain kernel proceeds as in Fig. 3. When all buffers of the new reverb tail are exhausted, certain partitions of the frequency domain kernel still contain the old reverb tail (502). In a succeeding compute cycle (504), each partition of the frequency domain kernel that contains the old reverb filter is filled with zeros. After several compute cycles (506), the frequency domain kernel contains only the new reverb tail, with zeros present at locations where the old reverb tail once resided. In a succeeding step, the partitions of the frequency domain kernel that contain zeros are removed by de-allocation of memory (508).

For a sound to be positioned external to the VR user, different, highly decorrelated reverberations need to be presented to each ear. To achieve this, a pair of decorrelation filters with a short number of taps, e.g., 600 taps, are used to create an impression of two long, e.g., 144,000-tap, filters from one FFT filter. Thus although just one FFT filter is created, per techniques of this disclosure, decorrelated reverberations are generated for presentation to each ear, resulting in savings on computational load. The decorrelation filters each have a cosine-modulated response with a time-delay between the cosine modulations. The decorrelation filters are such that their summed power-versus-frequency response differs very little (less than 0.1 dB) from unity. The decorrelation filters can be implemented using a sparse and efficient finite impulse response filter structure in the time domain.

Additionally, if the gain of a reverb tail is decreased, it can reach an inaudible level faster than the corresponding RT60 indicates. Thus, depending on gain of the reverb tail, a truncation of the reverb tail is performed to beneficially reduce computational load with no impact on user experience. The truncation is based on an adjustment factor, dependent on reverb tail length, that improves efficiency of the reverberation in cases where the gain is low.

CONCLUSION

Techniques of this disclosure enable the low-complexity production of realistic and externalized audio reverberations with smooth transition during a change in a virtual reality environment. The techniques enable such production with low computational load by precomputing and storing certain constituents that are needed to compute the reverb tail. Smoothness during transition of reverb tails is achieved by spreading the transition over several compute cycles. Sound externalization is achieved by presenting decorrelated reverberations to each ear. In this manner, realistic audio for VR can be produced on consumer devices with limited computational capability.