Three-Dimensional Audio Localization using Wavelet-Domain Convolution

Paul Hubbard[1], Kristin L. Umland[2], M. Cristina Pereyra[2], Nadine Miner[3], Thomas P. Caudell[1]

[1] *Department of Electrical and Computer Engineering*
[2] *Department of Mathematics*
University of New Mexico
and
[3] *Sandia National Laboratories*
phubbard@eece.unm.edu

Abstract

This paper describes a new approach for creating compelling virtual acoustic environments by synthesizing and three-dimensionally localizing sounds within the wavelet-domain. A prototype system was developed that couples this result with a new method for syntheszing parametrically controlled sounds in the wavelet domain [1]. Using the approach described in this paper, virtual environments may soon have a real-time method for creating realistic, dynamically controlled sounds that are localized in 3D space. Results are presented and discussed, with suggestions as to directions of further interest.

1. Introduction

The origins of this work lie in the field of virtual reality: in the quest for ever-increasing realism and fidelity, workers have begun to focus more upon the role of sound. As documented in [2] and [3], convolution of a sound source with a Head-Related Tranfser Function (HRTF) produces audio localized in a three-dimensional space for a human listener. In present systems, this convolution is typically done either in the time domain or via the Fast Fourier Transform (FFT). Because both approaches are computationally expensive, dedicated hardware is often required to achieve reasonable performance. Although computationally demanding, localized audio can add dramatically to the perceived 'presence' of a virtual environment.

Commercial products use abbreviated HRTF filter sets, reduced sampling rates, optimized convolution algorithms and other methods for HRTF localization of sounds in an attempt to keep costs down. While cost is less of an issue in research, we are still concerned with conservation of both CPU cycles and memory. For continuous (i.e. background) audio, a great deal of storage is required if we simply store large enough segments to avoid audible repetition.

Miner [3] produced a sound synthesis system capable of generating continuous audio from a small initial sample. The sounds can be parameterized to reflect changes in a virtual environment. For example, rainfall can easily be changed from light to heavy rain, with no perception of looping or discontinuities. The primary tool used was the wavelet transform. This method ameliorates the storage requirements for continuous sound, but introduces additional processing.

The solution we propose here for audio localization is to convolve in the wavelet domain rather than in the time domain or using the FFT. Sounds are synthesized using the [3] approach, and localization following by HRTF convolution in the wavelet domain. The main advantage of this approach is that only one inverse wavelet transform is required to transition directly to the final three-dimensional, time-domain audio environment. What is desired is a system as shown in figure one, where both synthesis and localization are done in one domain. In this paper, we introduce methods to perform the HRTF convolution in the wavelet domain. Benefits and drawbacks of this approach are explained, beginning with an exposition of the math

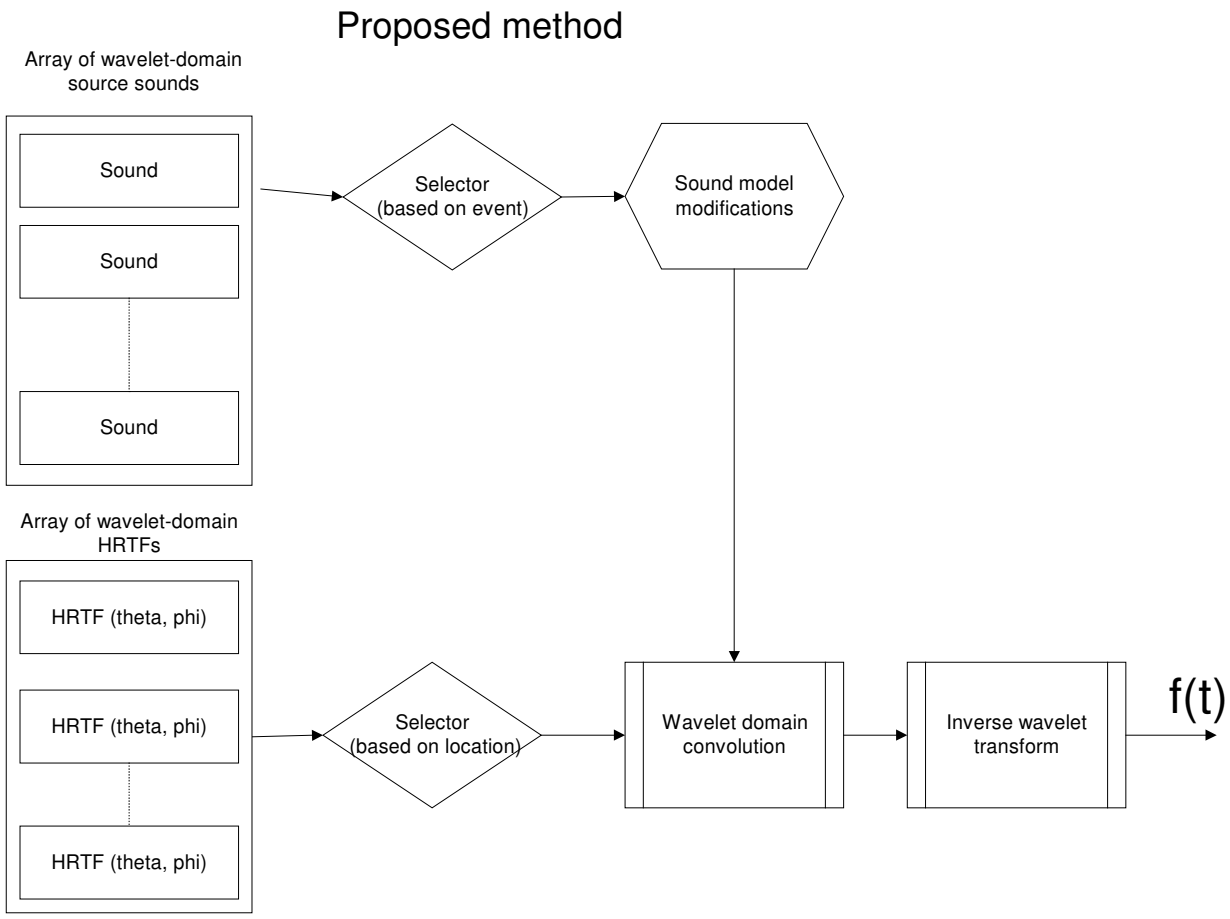

Figure 1: Flowchart of proposed method

background required. Results and discussion are then presented, with suggestions as to further avenues of interest.

2. Mathematical Background

The cornerstone of digital filtering is the convolution operator. Via convolution, every linear time-invariant operator can be applied to an input signal. For the purposes of this paper, an operator is defined to be a function acting on an vector, that puts that vector into another space. Examples include the Fourier transform, differentiation (yielding the gradient), and the Hilbert transform.

However, straightforward time-domain convolution requires $O(n^2)$ [5] operations, making it unsuitable for real time processing. The FFT, which runs in O(nlogn) [5], can be used to efficiently implement circular convolution, enabling fast block mode filtering. In the Fourier domain, circular convolution becomes multiplication, an elegantly simple operation. In other words, the Fourier transform diagonalizes all linear time-invariant operators [6]. It can be shown that the Fourier domain is the only domain where this is the case. [7,8]

From this initial result, we know that wavelet-domain convolution is necessarily less efficient. However, Beylkin has shown in [9] and [10] that the wavelet transform ‘nearly diagonalizes' all Calderon-Zygmund (i.e., similar to the Hilbert transform) and pseudo-differential (those similar to differentials) operators.

Beylkin introduces two methods of implementing operators: the non-standard form (of a matrix), and via filter banks. The non-standard form, or NSF, is explored in this work, and the filter bank left for further research.

2.1 The Non-standard Form

The non-standard form is primarily an approximation technique for applying an operator within the wavelet domain. While not exact, it can be made arbitrarily accurate by increasing the number of terms in the approximation. We can trade the accuracy of the result against the computational time required to calculate it. To understand how this is done, a brief digression into matrix forms is required.

The non-standard form is applied to the matrix form of an operator. This means that to implement convolution, we first write the convolution operator in the form of a circulant matrix. (The circulant nature arises because we are implementing circular convolution.) This is a Toeplitz matrix [11], where all diagonals are constant, and the rows wrap around. In the first column are the entries of the HRTF, in the second column the same entries are shifted down one place, with the last entry being wrapped around to the first row, and this pattern repeats. This matrix, applied to a vector, will convolve the vector with the HRTF. However, this multiplication is an inefficient operation of $O(n^2)$ [9], which is why filtering is not usually done via matrices. If we write the matrix in a wavelet basis, we obtain something like Figure 2, which is a schematic representation of a matrix, with the small-valued entries removed.

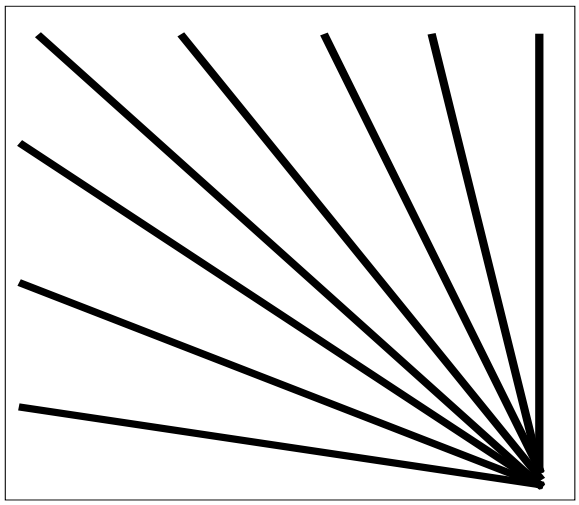

Figure 2 Schematic of truncated entries, matrix form of an operator matrix.

The fingerlike structure of this matrix arises from the interaction of different wavelet scale levels, and this shows that the different scale levels are not seperable.

By rearranging the standard matrix form into the NSF [10], we can decouple the interactions between scales and increase the potential efficiency of applying the matrix onto a vector. This requires keeping more information, albeit in sparse form, so the matrix area quadruples. Figure 3 shows the effect of the Non-Standard transformation. We have to keep track of a larger (albeit containing no more entries) matrix, but have succeeded in separating the various scales. Note also that rearrangement of the data vector required to apply this, because we have reorderd the scales within the matrix.

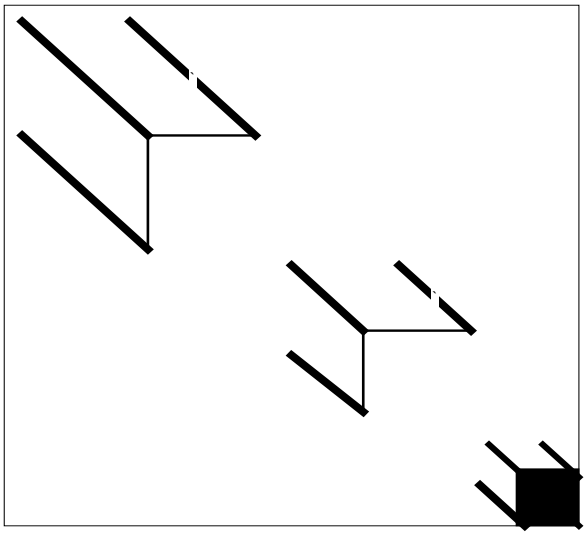

Figure 3: Schematic of truncated entries, non-standard form of a operator matrix

Figure 3 shows the sparseness of the NSF. Note that this transformation is a lossless operation. However, the next operation is to threshold the matrix, and this is the tradeoff between accuracy and storage requirements. In [10] Beylkin derives the expression for the time complexity of applying the NSF matrix onto a vector as $O((-\log \sigma)n)$, where sigma is the relative error, and n is the number of non-zero matrix entries. Since the scales are now decoupled, applying the NSF matrix onto a vector potentially becomes an O(n) operation. Note, however, that the wavelet-domain data vector must be permuted (e.g., via a lookup table) into the non-standard form. Since the sound synthesis operations are performed in the wavelet domain, the overall process simplifies nicely, as is shown in Figure 1.

2.2 The Filter Bank Method

After the development of the NSF, Beylkin published [12] results on the implementation of operators via filter banks. In comparison to the matrix method, filter banks are potentially more efficient for real time operation, and have the side benefit of being streaming instead of block mode operators. This is being pursued as the next step in the research program. The interested reader is referred to the Beylkin paper [12] for further details of the mathematics.

3. Implementation of the non-standard form

The lingua franca of the wavelet community is largely MATLAB [13]. In addition to its native signal-processing routines, several free and commercial toolkits exist for working with wavelets. Moreover, the Massachusetts Institute of Technology (MIT) Media Lab HRTF set [14] also has associated routines written in MATLAB for using its HRTFs. For these reasons, the tools used for this work are MATLAB and WaveLab [15].

3.1 Circular convolution and zero padding

The programs are implementing an approximation of circular convolution, and we must therefore be careful in dealing with wraparound effects that this causes when the input signal is longer than the HRTF. Following the approach of [16] and [5], we padded the HRTF with zeros, and did additional zero padding at the start and end of the sound file to ensure that all wraparound-distorted data was discarded. This increases the size of the NSF matrix to 4096 by 4096, but given its highly sparse nature it did not prove to pose a storage burden.

For information on source code, please visit our website [17], which also contains the sound files and other implementation details.

4. Results and discussion

In this section we give the results of numerical experiments with MATLAB programs. The purpose of these tests are to quantify the computational complexity of the localization algorithms, and to qualitatively assess the resulting sound quality. Human subjects experiments, similar to [18] are currently being designed to quantify the assessment.

All results are measured using MATLAB's internal ‘flops’ counter, and should be regarded as qualitative rather than exact. We assume that these results provide reasonable estimates of the relative complexity of the implementations.

Epsilon, the error threshold, is as used in the WaveLab [15] code. It is defined in terms of the maximum row norm of the NSF matrix, where anything less than epsilon times the maximum row norm is discarded. We assume that the maximum row norm was chosen due to its invariance over the orthonormal bases. In simple terms, epsilon is how we choose entries to erase or keep. The choice of a different norm is another item of future work.

4.1 Non-Standard Form Results

In Table 1, the first column is the wavelet, the second column is the truncation threshold, as discussed above. The third column shows how many entries in the NSF matrix are non-zero, expressed as a percentage of the total 4096 by 4096 matrix. The megaflops (Mflops) are as measured by MATLAB, and the last column is a quick, subjective judgement of the perceived quality. For this table, only an audio file of piano music [17] was used.

Table 1: Timings and Sparseness Degrees for Wavelet-Domain Localization

Wavelet	Epsilon	Percent non-zero	Mflops	Quality
Daub.4	0.0001	9.80	868.5	Excellent
Daub.4	0.0002	8.33	744.1	Passable
Daub.4	0.0004	6.05	551.4	Poor
Daub.20	0.0001	7.83	842.2	Excellent
Daub.20	0.0002	5.79	669.7	Passable
Daub.20	0.0004	4.02	520.0	Poor
Coiflet 5	0.0001	7.76	928.7	Excellent
Coiflet 5	0.0002	5.72	756.5	Passable
Coiflet 5	0.0004	4.04	614.1	Poor
Beylkin	0.0001	7.23	765.0	Excellent
Beylkin	0.0002	5.33	607.1	Passable
Beylkin	0.0004	4.34	528.6	Good

4.1.1 Efficiency and Compression

By way of comparison, the MATLAB-based routine utilizing convertional convolution took approximately 52.9 megaflops to perform the same HRTF convolution. Even at its best, the NSF approach is at least ten times more complex in time, and a thousand times more storage-intensive.

Some wavelets, notably the Beylkin wavelet, are slightly more effective at compression, but the difference is relatively slight. It was expected from [12] that operator compression would scale with the number of vanishing moments of the wavelet function; while true (e.g. Daubechies-20 versus Daubechies-4) there is not a large difference.

Even at epsilon equaling 0.0004, we still have a great deal more information than exists in the original HRTF filter. The matrix is compressed from the original circulant form, but still requires much more memory than the original: the number of non-zero entries is roughly three orders of magnitude greater.

4.1.2 Subjective Evaluations

Informal auditioning of the results indicates that the difference between normal and wavelet-domain convolution is discernable when epsilon is less than about 0.0002, but *only* in an A/B comparison. Subjectively, the quality of the approximation is excellent, but there are subtle differences from the true convolution that are discernable in a direct comparison. As epsilon increases, an audible buzz is introduced into the result, with irritating results. Rigorous perceptual testing will be conducted to determine an acceptable error threshold and this would probably vary with each application's error tolerance and audience.

We found that audio containing only piano to be the most useful. Subjectively, it was most revealing of reconstruction errors. An interesting related question is that of reference audio samples, which will be pursued in future work.

4.2 Synthesis Operators

Once we had obtained working wavelet-domain convolution, we implemented a version of the Miner and Caudell [3,1] sound synthesis operators into the code.

The essence of the basic synthesis operator is the multiplication of a block of wavelet coefficients at a given scale by a fixed scalar. To implement this , the localized results are multiplied by a diagonal matrix, where most entries are unity and the magnitude scale factor is placed according to the scale one wishes to modify. By making the matrix sparse, this adds a relatively small amount of computation and storage, though it is obviously much less efficient than it could be. It is, however, flexible and quick to implement.

Since the synthesis operators work better with sounds with high stochastic content, such as rainfall, we found a recording of a stream that provided good results. We have yet to perform rigorous experimentation with these operators, but initial ad hoc experimental results were congruent with those of Miner and Caudell. For example, modifying fine detail coefficients added random-seeming high frequency content, and the results did not sound 'artificial.'

5. Conclusions and Further Work

We have successfully verified the application of [10] to audio localization. We have also shown that the sound synthesis technique of Miner and Caudell works in different bases than originally tested. Addtionally, we now have a solid mathematical understanding of the processes involved in both localization and synthesis. However, we did not obtain good compression or time complexity with the matrix form of these algorithms.

In a larger setting, however, the lack of efficiency is of less interest than the validation of the method. We can now easily perform perceptual experiments with localization, synthesis operators, and search the parameter space thus provided to find perceptually useful operations and parameters. Efficiency, while quite nice, can be obtained by a number of related techniques, foremost of which is the filter bank [12].

Further work is therefore planned to implement the filter bank method, with the eventual goal of a software-based synthesis and localization system that can run on different desktop machines in tandem with a virtual reality simulation.

It is of note that we have left unexplored the vast territory of other operators in the wavelet domain. We know from [10] that the entire Calderon-Zygmund class of operators, as well as the pseudo-differential operators, can be implemented by these methods. Quite naturally, the question arises: what else can we use in this rich operator class?

In summary, we have successfully implemented wavelet-domain convolution, have a mathematical foundation for other operators [9,10,12] and further work in filter banks. The prototype system enables experimentation with various basis functions, error thresholds, synthesis operators, input files, and audio positioning. It looks promising that these developments, along with new wavelet-based sound synthesis techniques, could lead to an integrated, cost-effective, real-time virtual sound server. However, computational efficiencies are poor, with operator compression lagging behind expectations and needing further work. Extending these results to real-time will require a different implementation. Future work will focus upon implementation of a real time system, further integration of the synthesis algorithms, and perceptual validation of the results.

6. Acknowledgments

The authors wish to thank the University of New Mexico Albuquerque High Performance Computing Center and the Dept. of Electrical and Computer Engineering for providing resources for this research, as well as the authors of the WaveLab toolkit. Additionally, the authors thank B. Gardner, K. Martin and the MIT Media Lab for their collection of HRTFs and associated MATLAB routines.

7. References

1. Miner, N. E. and Caudell, T. P. A Wavelet Synthesis Technique for Creating Realistic Virtual Environment Sounds. Submitted for publication to Presence: Teleoperators and Virtual Environments, 1999.

2. Wenzel, E. M.. Localization in virtual acoustic displays. *Presence: Teleoperators and Virtual Environments, Vol. 1, No. 1,* 1992, pp. 80-107.

3. Miner, N. E.. Creating Wavelet-based Models for Real-time Synthesis of Perceptually Convincing Environmental Sounds. Ph. D. dissertation, University of New Mexico, 1998.

4. D. Begault, *3-D Sound for Virtual Reality and Multimedia*, Academic Press, 1994.

5. W. Press, S. Teukolsky, W. Vetterling, *Numerical Recipes in C*, Cambridge University Press, 1996.

6. S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, 1998.

7. M. Frazier, *An Introduction to Wavelets Through Linear Algebra*, Preprint, 1999.

8. G. Kaiser, *A Friendly Guide to Wavelets*, Birkhauser, 1994.

9. G. Beylkin, R. Coifman, and V. Rokhlin, *Fast Wavelet Transforms and Numerical Algorithms I*, Communications in Pure and Applied Mathematics, Vol. 44, no. 2, 1991, pp. 141-183.

10. G. Beylkin, *Wavelets, Multiresolution Analysis and Fast Numerical Algorithms*, A draft of INRIA lectures, ftp://amath.colorado.edu/pub/wavelets/papers/index.html, 1991.

11. N. Young, *An Introduction to Hilbert Space*, Cambridge University Press, 1988.

12. G. Beylkin and B. Torresani, *Implementation of operators via filter banks, autocorrelation shell and Hardy wavelets*, Applied and Computational Harmonic Analysis, 3, 1996, pp. 164-185.

13. MATLAB, version 5.1, MathWorks Inc., http://www.mathworks.com/

14. B. Gardner and K. Martin, *HRTF Measurements of a KEMAR Dummy-Head Microphone*, MIT Media Lab, http://sound.media.mit.edu/KEMAR.html

15. D. Donoho, X. Huo, T. Yu, *WaveLab v0.7*, http://playfair.stanford.edu/~wavelab

16. G. Strang and T. Nguyen, *Wavelets and Filter Banks*, Wellesley-Cambridge, 1996.

17. P. Hubbard, K. Umland, C. Pereyra, http://www.arc.unm.edu/~phubbard/wdconv/

18. Bonebright, T., Miner, N., Goldsmith, T., Caudell, T. *Data Collection and Analysis Techniques for Evaluating the Perceptual Qualities of Auditory Stimuli*. Proceedings of the International Conference on Auditory Displays, Glasgow, Scotland, 1998.