MPV: SIMULATING MANY PHASE VOCODERS

Jaime E Oliver La Rosa

Columbia University
Department of Music - CMC, New York City, USA
jo2357@columbia.edu

ABSTRACT

This paper presents *mpv* or "many Phase Vocoders"; a simple way of modifying a Phase Vocoder to simulate the use of several Phase Vocoders at a lower computational cost. It then explores its possible and current applications as well as its benefits and weaknesses.

1. INTRODUCTION

A Phase Vocoder is an FFT-based technique for time scaling and pitch transposition of a fixed or recorded sound. It was originally developed for speech encoding. While it is a powerful technique, it is also computationally expensive. As a consequence, the use of multiple instances of this technique in live computer music performance is limited by computational power. *mpv* or "many phase vocoders" is a simple modification to the phase voco-der algorithm that allows, given certain considerations, to simulate the use of several phase vocoders at a lower computational cost.

1.1. Outline

The remainder of this article is organized as follows. Section 2 gives an account of previous work on which *mpv* builds upon and presents the changes to the original technique used in *mpv*. Sections 3 and 4 present the computational improvements versus the conditions where this technique might not be ideal. Finally, Section 5 presents a few cases where the algorithm has been applied successfully and what some potential uses might be.

2. PREVIOUS WORK AND MODIFICATIONS

Originally proposed by Flanagan [4] and implemented as an FFT technique by Portnoff [8], the phase vocoder was later used for computer music applications by [5] and [3]. This technique is developed on the basis of Puckette's implementation [9] in the Pure Data Environment [10]. *mpv* is based on the I07.phase.vocoder.pd example patch of the Pure Data distribution, version 0.43.

A representation of the original phase vocoder (or *pv*) is shown in Figure 1. It can be divided into two big processes: *reading point* and *analysis-synthesis*.

The *reading point* process feeds the *analysis-synthesis* process two overlapping windows of a sound of which one is 1/4 of a window behind therefore having an overlap

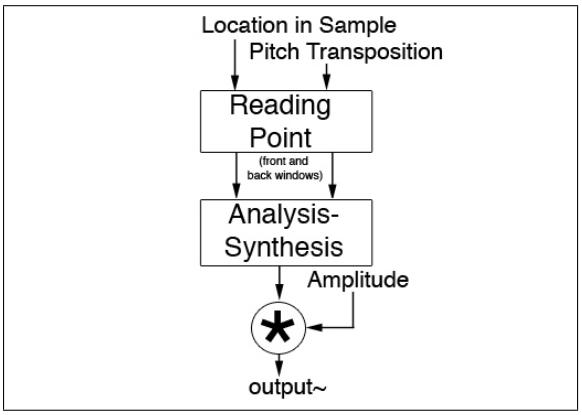

Figure 1. Design of a phase vocoder.

of 3/4 of a window. These two overlapping windows are known as "front and back windows". The *reading point* process allows for the control of the location in the sample and the pitch transposition. The *analysis-synthesis* process finds the change of phase between these windows and adds it to the phase of the previous analysis.

In *mpv* both the *analysis-synthesis* and the *reading point* processes are left intact, however, there are several instances of the latter. In other words, there are many independent reading points. Their output signals are added, multiplied by an amplitude control and sent to the *analysis-synthesis* process. The process is shown in Figure 2.

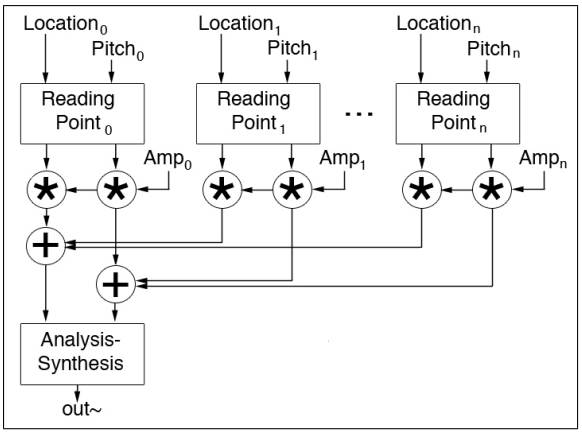

Figure 2. *mpv* process layout.

3. RESULTS

To compare these results, the cpu consumption of each of these processes was measured. The measurements were taken in a mac book pro computer with a 2.26Ghz, 3Mb L2 Cache, Intel Core 2 Duo processor running OS X 10.6.8 11. Both *pv* and *mpv* use a window size of 2048 points and an overlap of 4. In figure 3., the vertical axis represents cpu consumption and the horizontal axis represents the number of instances of additional phase vocoders versus additional reading points within one *mpv*.

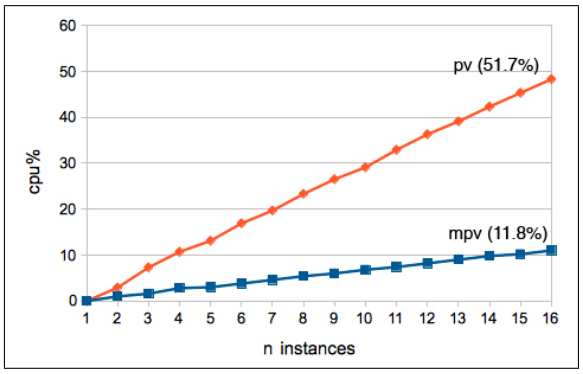

Figure 3. Marginal cpu consumption for *mpv* and *pv*

In the origin, we have one full phase vocoder with one *reading point* and one *analysis-synthesis* process. The *pv* curve reflects the marginal cpu increase for every new *pv* instance - that is, for the increment of both *reading point* and *analysis-sysnthesis* processes for each instance. The *mpv* curve reflects the marginal cpu increase for every new *reading point* process within one *mpv*.

As it can be clearly seen, there is a substantial cpu process difference that can be used to either have more power for other processes that might have to happen at the same time or have more reading points in an *mpv* than instances of emphpv's that a cpu can handle.

4. CONSIDERATIONS

4.1. A Sinusoidal Model

Lets imagine we have two sounds that are sweeping sinusoids. In a first case, we have these two sounds in the same audio file and therefore can only be modified together. In a second case, we have them in separate files and can be modified independently. The principle behind *mpv* is that of additive synthesis: the *analysis-synthesis* process does not care if the sounds it receives are being read from a pre-mixed (or pre-added) file or read from separate files and added right before the analysis as in the case of *mpv*.

Phase Vocoders are based on sinusoidal models of sound. As such, noisier sounds are harder to transform without noticeable artifacts, because the *analysis-synthesis* process is trying to fit a non-sinusoidal sound into a sinusoidal model. In this sense, both the *pv* and *mpv* present a limitation.

Harmonic sounds, of not very low pitch, are easier to stretch and transpose because sinusoidal components tend to be at a reasonable distance from each other so as to not cause any conflict, given adequate window sizes and sample rate. Increasing frequency resolution decreases time resolution and vice versa. Dolson [3] explains it in the following way:

> "[a phase vocoder] fails when there is more than a single partial in a given band, or when the time-varying amplitude or frequency of the bandpass signal changes too rapidly. 'Too rapidly' means that the amplitude and frequency are not relatively constant over the duration of a single FFT"

A potential problem of using *mpv* is that by adding several sounds harmonics form different sounds might end up too close to each other, rendering the model incapable of finding a good solution within the sinusoidal model. For example, the *analysis-synthesis* process is incapable of re-synthesizing beating and results in audible artifacts.

In other words, as sounds get more complex, the phase vocoder is less capable of discerning the individual sinusoidal components. This is true of both the *pv* and *mpv*, however the *mpv* is more likely to reach this problem as it deliberately adds complexity to the sound to be analyzed.

Considering these issues, one needs to decide whether sounds are added into an *mpv* or treated separately in different *pv*'s and added after the *analysis-synthesis* process.

On the other hand, while "artifacts" might emerge, it is important for the composer/ author to determine whether these are intolerable, tolerable or perhaps a welcome outcome after all.

If these results were to be deemed undesirable, the technique can still be useful if the composer chooses to work with sounds that do provide desirable results. I have worked with this technique already and do not complain. However, when attempting to re-create these artifacts feeding the algorithm less than ideal sounds, then they are clearly audible.

4.2. A Single Mixed Output

As it is to be expected, because sounds are pre-mixed before the analysis process, the output is also a mix. This does not allow for independent spatialization and or further independent processing after the *analysis-synthesis* process. However, each reading point can be sent to several *analysis-synthesis* processes. Controlling the amplitudes of these reading points might be used to control spatial processes.

5. APPLICATIONS

The *pv* was effectively used in Wishart's composition *Vox 5* [12] and Bailey et al.[1] have suggested its use for instrument design. I have used *pv*'s in several of my compositions with the Silent Drum [7] and MANO controllers

[6]. However, this technique is computationally expensive and its use in large numbers can easily overflow the computational resources of a system. *Mpv* was developed as a means to achieve similar results with less computation.

Mpv can be used in several situations depending on the context. I have successfully used this algorithm in the composition of new works including *9 gardens* for MANO Controller.

It has already been used successfully in Roger Reynolds *SMEARZ* algorithm by Roger Reynolds, as implemented by the author [11] where it is used to achieve the simultaneous stretching of several parts of the same sound. This algorithm is used in the works *Seasons*, *Dream Mirror* and *Marked Music*. This implementation uses 64 reading points and 3 outputs.

I am currently working on using this technique to interpolate between grains classified and ordered with Brent's timbre analysis toolkit for Pure Data [2], achieving timbal navigation of a multidimensional space.

The code and audio examples are available at the following address: www.jaimeoliver.pe/mpv

6. CONCLUSIONS

The modifications done to a phase vocoder or *pv* here called *mpv*, can provide the illusion of having many phase vocoders at a lower computational cost. Depending on the sound material being transformed, this reduction in computation may cause artifacts. These artifacts might not necessarily be undesirable or perceivable.

7. REFERENCES

[1] N. Bailey, A. Purvis, I. Bowler, and P. Manning, "Applications of the Phase Vocoder in the Control of Real-time Electronic Musical Instruments," *Journal of New Music Research*, vol. 22, no. 3, pp. 259–275, 1993.

[2] W. Brent, "A Timbre Analysis and Classification Toolkit for Pure Data," in *Proceedings of the International Computer Music Conference*, 2010, pp. 224–229.

[3] M. Dolson, "The Phase Vocoder: A Tutorial," *Computer Music Journal*, vol. 10, no. 4, pp. 14–27, 1986.

[4] J. Flanagan, "Phase Vocoder," *The Bell System Technical Journal*, pp. 1493–1509, 1966.

[5] J. Moorer, "The Use of the Phase Vocoder in Computer Music Applications," *Journal of the Audio Engineering Society*, vol. 26, no. 1/2, pp. 42–45, 1978.

[6] J. Oliver, "The MANO Controller: A Video Based Hand Tracking System," in *Proceedings of the International Computer Music Conference*, 2010.

[7] J. Oliver and M. Jenkins, "The Silent Drum Controller: A New Percussive Gestural Interface," in *Proceedings of the International Computer Music Conference*, 2008.

[8] M. Portnoff, "Implementation of the Digital Phase Vocoder using the Fast Fourier Transform," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 24, no. 3, pp. 243–248, 1976.

[9] M. Puckette, "Phase-Locked Vocoder," in *Applications of Signal Processing to Audio and Acoustics, 1995., IEEE ASSP Workshop on*. IEEE, 1995, pp. 222–225.

[10] ——, "Pure Data: Another Integrated Computer Music Environment," *Proceedings of the Second Intercollege Computer Music Concerts*, pp. 37–41, 1996.

[11] R. Reynolds and J. Oliver, *SEASONS: Technical Score*. Edition Peters - C. F. Peters Corporation No. 68300i, New York, USA, 2010.

[12] T. Wishart, "The Composition of "Vox-5"," *Computer Music Journal*, vol. 12, no. 4, pp. 21–27, 1988.