Real-time FOF and FOG synthesis in MSP and its integration with PSOLA

Michael Clarke[1] and Xavier Rodet[2]

[1]Department of Music,
University of Huddersfield,
Queensgate, Huddersfield, England, HD1 3DH
j.m.clarke@hud.ac.uk

[2]IRCAM,
1 place Igor-Stravinsky,
75004 Paris, France
rod@ircam.fr

Abstract

This paper presents new objects for MSP that provide highly flexible FOF and FOG generators in a real-time environment. These objects combine many of the features of earlier FOF/FOG objects for MSP with additional features, some of which were also part of the Csound implementation. The availability of these generators in MSP permits sophisticated and complex real-time control of aspects of this synthesis method at a level not available previously. Further work has been undertaken to link this work with PSOLA (Pitch Synchronous Overlap Add), permitting continuous transformation all the way between FOG synthesis and PSOLA synthesis resulting in more powerful and realistic transformations. Example patches illustrating these various features will be presented

1. Introduction

FOF synthesis is a complex and computationally intensive algorithm. When first devised and used for sound synthesis it was virtually impossible to run in real time. In 1988 Clarke and Rodet hand-coded an array processor and achieved real-time voice synthesis only with some difficulty. More recently radical increases in computing power have meant that off-the-shelf computers are able to deliver quite complex FOF syntheses without problem. The purpose of the current project has been to take advantage of this situation by making flexible FOF and FOG (granulation of sound using the FOF envelope) objects available which combine many of the features of earlier versions in the context of the real-time performance oriented environment of MSP. Linking the FOG object with PSOLA analysis data is a significant new development in connecting these two closely related approaches and permitting creative interaction between them to be explored.

2. Background

FOF synthesis was originally devised by Xavier Rodet (Rodet, 1984) when researching into speech synthesis and was incorporated into the IRCAM program 'CHANT' (Rodet, *et al*. 1984) to simulate the human singing voice and other timbres. Later a unit-generator for FOF synthesis was incorporated into Music 11 (and later Csound) by Michael Clarke (Clarke, *et al*. 1988) with the idea of using this method in a more experimental way, making full use of the flexibility of a modular software system to explore the granular nature of FOF synthesis and transitions between timbres and textures. Eckel, Iovino, and Dudas added a FOF object to MSP in 1998 together with a collection of rule-based objects to enable many of the control features of the original CHANT program to be realised easily in MSP.

FOG synthesis was proposed by Clarke as an extension of the concept of FOF, using sound samples in place of synthesised waveforms (usually sine waves) to perform a particular type of sound granularisation. Gerhard Eckel produced a version of this for MSP (Eckel, *et al.* 1995) and Clarke added a FOG unit-generator to Csound (Clarke 1996) using a somewhat different approach.

Recently Clarke, working with Rodet, has added FOF and FOG objects to MSP which combine many of the features of the Csound unit-generators with certain aspects of the earlier MSP objects. The FOG object has also been adapted to read data from PSOLA analysis files and therefore is able to offer PSOLA synthesis (Peeters and Rodet 1999, Schnell *et al*. 2000) and continuous transformation right from FOG synthesis all the way to PSOLA synthesis.

PSOLA analysis has been ported on Linux and Mac OSX at IRCAM by G. Peeters and J. Escribe and is available from the IRCAM Forum. PSOLA analysis/re-synthesis allows very high quality time stretching/compression, pitch modification and even spectrum stretching/compression.

3. The fofb~ object in MSP

Since FOF synthesis has now been widely used for a number of years the basic features are generally well known. New features of this MSP objects include the generation of a bank of FOF generators from a single object and spatialisation options.

Fofb~ produces a bank of formants and has as one of its arguments the number of formants to be synthesised. A single object then produces and sums the specified number of formant regions, with a common fundamental frequency but with independent parameters for each formant region (formant frequency, bandwidth etc). Such an approach makes the creation of patches for complex timbres easier and avoids the need for recalculating data shared between the different formants (e.g. fundamental phase). There are times however when it is still important to use a different generator for each formant, for example if one wants to be able to dissolve a single timbre into different grain streams or vice versa.

The spatialisation option in fofb~ permits the output of 1,2,4 or 8 channels of sound (specified by one of the arguments). Each grain is given a fixed location for its duration. In many granular contexts this striated spatialisation results in a richer textures than smooth movements as has been found previously by others working with granular synthesis such as Barry Truax.

Both the fofb~ and fog~ generators revert to the CHANT approach to input of envelope data (*tex, debatt, atten,* where *debatt* is the start of the attenuation, unlike the Csound unit-generators where *debatt* was replaced with *dur* – the duration of the whole envelope. Other features of the fofb~ generator include multiple octaviation (as found in the Csound FOF unit generator).

Examples of the fofb~ object in use. A key feature of the new object is the way in which a flexible FOF generator can be controlled in real-time as the demonstration will show, illustrating not only the basic parameters and CHANT examples but also granular timbral and textural transformations.

4. The fog~ object in MSP

This object corresponds closely to the Csound FOG unit-generator but includes spatialisation as described in relation to the fofb~ object above. This is similar to the spatialisation used by Eckel. However, this FOG object differs from Eckel's MSP object in that it can be driven by an in-built phasor signal (as is Clarke's Csound unit-generator) as well as by triggering messages. This facilitates use of the object for precisely synchronised streams of excitations as well as for individual grains or statistically determined grain clusters.

With the fog~ object a pre-recorded sound stored in a buffer is granulated using the FOF envelope (Clarke 2000). Data from the stored sound replaces the synthesised sine wave used in FOF. The speed at which the object reads through the buffer can be altered as can the speed at which the data for the grain waveform is read. In this way the speed and pitch of the original sound can be varied independently. FOG therefore has much in common with other forms of granular processing although the smooth FOF envelope reduces some of the artefacts produced by methods using, for example, simple linear envelopes.

As with FOF it is possible to produce both timbral and textural effects and this is particularly expressive with real-time control. In this case the timbral effects originate from the pre-recorded sound. Textural effects make more radical transformations, often by layering grains from the original to create rich textures combined with striated spatialisation. The possibility of moving continuously between the two approaches offers particularly interesting creative potential.

Examples of fog~ in use. The demonstration will show the basic operation of the fog~ object, its arguments, parameters and messages. It will illustrated the timbral and textural transformation of sound in real-time and the movement between this and timbral processing.

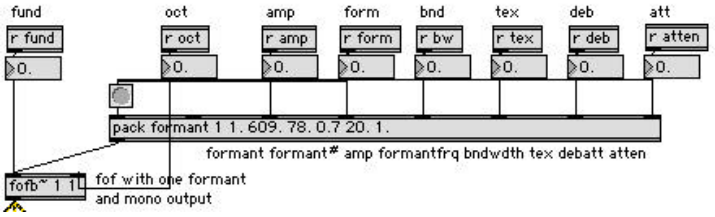

Figure 1: Part of the fofb~ help patch

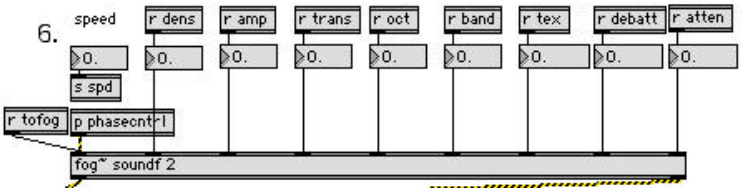

Figure 2: Part of the help patch for fog~

5. fog~ and PSOLA

A new development linking FOG with PSOLA analysis/re-synthesis further extends the creative potential of this approach, allowing PSOLA synthesis in the same framework and permitting continuous transformation without break between FOG synthesis and PSOLA synthesis.

With the normal fog~ object no account is taken of the content of the buffer in the process of granularisation. Grains are triggered either by the in-built phasor or by one of a number of trigger messages and whilst the speed of progression through the buffer can by controlled the precise position in the buffer from which grains are created is independent of the sound in the buffer. PSOLA analysis (Peeters and Rodet 1999, Schnell, *et al*. 2000) detects certain features about a sound file and stores this analysis data in an SDIF formatted file. It estimates where fundamental periods commence and for each period provides a time-tagged frame of data. This comprises (1) the position in the buffer in samples, (2) the fundamental frequency, (3) a voiced/unvoiced flag, (3) a coefficient for voiced/unvoiced, (4) a voiced/unvoiced cut-off frequency, (5) the length of the period in samples, (6) a modulation factor, (7) a transience flag.

A modified FOG object which takes account of this data can do PSOLA re-synthesis and provide more realistic transformations of sounds producing fewer side effects. Most importantly the positioning of FOGs can be aligned with the fundamental periods of the original signal. The normal FOF envelope is used with the peak coinciding with the start of a period (a single FOG in fact spans two fundamental periods). Figure 3 shows how the parameters of a FOF envelope can easily be set to produce a PSOLA envelope: (a) a typical FOF envelope, (b) the same with zero bandwidth (no exponential decay) and (c) *tex*, *debatt* and *atten* set to produce the symmetrical PSOLA envelope. The rise portion of the envelope begins at the start of the previous fundamental period and the decay (‘*atten*’) lasts from the start of the fundamental period to the start of the next exactly in the way PSOLA synthesis does. Determining the grains in this way permits time stretching to be performed without many of the side-effects normally experienced with granular processing. Where time-stretching results in a new grain being triggered between analysis frames interpolation is used.

Furthermore, other data in the PSOLA analysis can be used to improve the quality of the transformation. The voiced/unvoiced flag can be used so that unvoiced portions of the sound are processed (e.g. reversing alternate waveforms) to minimise the side effects normally associated with stretching unvoiced signals. The voiced/unvoiced frequency data indicates the cutoff frequency above which a voiced sound becomes unvoiced. By dynamically filtering the

sound prior to FOG processing into two files, voiced and unvoiced, it is again possible to process the unvoiced portion of the signal appropriately.

The modulation factor may be used, in conjunction with the fundamental frequency to remove vibrato before processing and then re-impose it afterwards avoiding unnatural changes to vibrato speed resulting from time expansion/compression. The transience flag may be used to eliminate processing of those portions of the signal which are inappropriate.

In fact remarkably little change has been needed to the fog~ object to adapt it for use with PSOLA. The essential FOG algorithm remains unchanged and it is simply a question of using the PSOLA data to control the timing and parameter settings for the generation of FOGs. This also makes it very easy to move between ordinary FOG operation and PSOLA-controlled FOGs. By moving between these two options interesting transitions can be produced interpolating between realistic transformations and transformations which deliberately introduce new artefacts into the sound. Working with this in the context of MSP means that real-time control can be very flexible as will be demonstrated.

6. Future developments

The fofb~ and fog~ objects have been completed and were released on the October 2002 IRCAM Forum CD together with help files and a number of demonstration patches. The PSOLA version is currently (Spring 2003) operational in a prototype form and undergoing further testing and development. It is intended that it be available for release by Autumn 2003.

(a) **(b)** **(c)**

Figure 3: From FOF envelope to PSOLA envelope

References

Clarke, J. Michael, P. D. Manning, R. Berry, and A. Purvis 1988. "VOCEL: New implementations of the FOF synthesis method." *Proceedings of the 14th International Computer Music Conference, Cologne, 1988,* 357-71.

Clarke, J. Michael, 1996. "TIM(br)E: Compositional approaches to FOG synthesis" *Proceedings of the International Computer Music Conference, Hong Kong*, 1996 375-7.

Clarke, J. Michael, 2000. "FOF and FOG Synthesis in Csound." *The Csound Book*, ed. R. Boulanger, MIT Press, 2000, 293-306.

Eckel, Gerhard, M. R. Iturbide, and B. Becker 1995. "The development of GiST, a granular synthesis toolkit based on an extension of the FOF generator." *Proceedings of the International Computer Music Conference, Banff, Canada, 1995,* 296-302.

Peeters, Geoffroy and X. Rodet 1999. "SINOLA: A new analysis/synthesis method using spectrum peak distortion, phase and reasssigned spectrum." *Proceedings of the International Computer Music Conference, Beijing, 1999*, 153-156.

Rodet, Xavier,.1984. "Time-domain formant-wave-function synthesis." *Computer Music Journal* 8.3 :9-14.

Rodet, Xavier, Y. Potard, and J.-B. Barrière,. 1984. "The CHANT Project : From the synthesis of the singing voice to synthesis in general." *Computer Music Journal* 8.3 :15-31.

Schnell, Norbert, G. Peeters, S. Lemouton, P. Manoury and X. Rodet 2000. "Synthesising a choir in real-time using Pitch Synchronous Overlap Add (PSOLA)." *Proceedings of the International Computer Music Conference, Berlin, 2000,* 102-108.