FACTORSYNTH: A MAX TOOL FOR SOUND ANALYSIS AND RESYNTHESIS BASED ON MATRIX FACTORIZATION

Juan José Burred
Paris, France
`jjburred@jjburred.com`

ABSTRACT

Factorsynth is a new software tool, developed in the Max environment, that implements sound processing based on matrix factorization techniques. In particular, Non-negative Matrix Factorization is applied to the input sounds, which produces a set of temporal and spectral components that can be then freely manipulated and combined to produce new sounds. Based on a simple graphical interface that visualizes the factorization output, Factorsynth aims at bringing the ideas of matrix factorization to a wider audience of composers and sound designers.

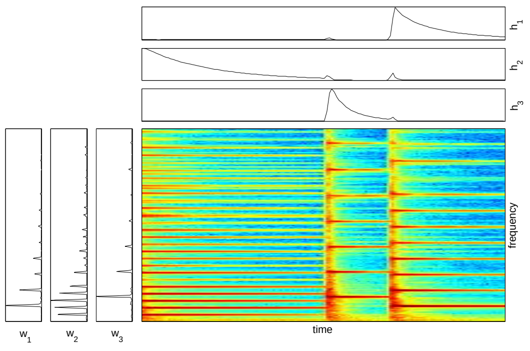
1. INTRODUCTION

Any kind of data in matrix form can be subjected to factorization, i.e., to an algorithm that yields two or more output matrices (the factors) which, when multiplied back together, produce an approximation of the input. There is a wide range of factorization algorithms that can produce very different factor matrices, depending on the constraints imposed by the desired application. By analyzing the resulting factor matrices it is possible to discover and separate important underlying components, often called *latent variables*, that were hidden and mixed within the original data. Because of this, factorization is central to many computing fields such as data compression, computer vision or machine learning.

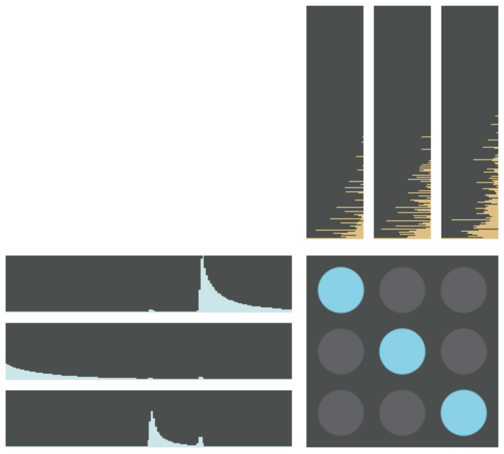
In the audio domain, matrix factorization is most often applied to the magnitude or power spectrogram, which is a matrix whose rows are time-varying energies of individual frequency bands, and whose columns are spectra at given times. One of the most widely used factorization algorithms in sound applications is Non-negative Matrix Factorization (NMF) [1], which imposes the constraint that all elements of the input and output matrices have to be zero or positive. NMF results in two factor matrices, one containing spectra (called *bases*) and the other containing temporal functions (called *activations*). The combination of one of the spectral bases with its corresponding activation results in a *component sound*, an approximation to a sonic entity or event contained in the input signal. Component sounds can be, for instance, individual notes, drum hits, or any kind of temporally or spectrally distinctive event.

NMF is thus ideal for applications that require analyzing or resynthesizing separate elements of the input sound, and is currently widely used in fields such as source separation or music information retrieval. An illustration of a simple sound decomposed by NMF is shown in Fig.1(a). The input sound (spectrogram shown) is a sequence of 3 piano notes of different pitches. The output of the factorization are a set of 3 spectral bases (displayed at the left of the spectrogram) and a set of temporal activations (displayed on top). It can be seen that the peaks in the activations correspond to the temporal positions of the corresponding spectral bases in the input sound.

The application of matrix factorization to musical cre-



(a) Input spectrogram with extracted spectral bases (left) and activations (above)



(b) Display convention used in Factorsynth

Figure 1. Visualization of a simple sound (three-note piano melody) decomposed by NMF.

ation is fairly recent [2–5]. As a sound decomposition method, it fits well into the analysis/resynthesis paradigm of computer and electronic music, in which a sound is modified by manipulating parameters resulting from its previous analysis (or, in cross-synthesis, from the analysis of a second sound). In traditional additive analysis/resynthesis (phase vocoder), the decomposition bases are sinusoids. In this context, matrix factorization can be seen as a higher-abstraction version of additive analysis/resynthesis in which each sinusoid has been replaced by a full spectrum.

A framework for performing sound modifications and cross-synthesis based on factorization was presented in [6]. This article presents a graphical implementation thereof in the Max environment. The basic principle of Factorsynth is the ability to freely recombine any spectral basis with any temporal activation resulting from factorization. In other systems aimed at analysis and separation, each basis always remains coupled with its corresponding activation, since the goal is to reconstruct elements that are actually present in the original sound. In contrast, here the goal is to create new sounds, and so arbitrary recombinations are allowed.

A preliminary implementation was presented in the form of a command-line executable [6], which was of limited usability and control capabilities. The new Max version provides a graphical interface and thus the possibility for the user to visualize the result of factorization, listen to the separated components, edit the extracted bases and activations, and closely control the resynthesis process. The ability to freely perform any base/activation combination is emphasized by the central element of the graphical interface: a switchboard that symbolizes the couplings between bases and activations. An example of the Factorsynth visualization of factorization and recombination is shown in Fig.1(b), which corresponds to the same three-note piano sound of Fig.1(a). For easier alignment, the spectral bases are displayed above the switchboard, and the temporal activations to its left. An activated button on the switchboard means that the basis above it and the activation to its left are to be combined for resynthesis. In the figure example, the switchboard has its diagonal elements activated, which means that in this case resynthesis will approximate the original sound without modifications.

Factorsynth is freely available for download [1]. Several sound examples are also presented in the download page.

2. THE FACTORSYNTH INTERFACE

A number of controls are available in the interface, together with the display of the components and of the switchboard. Two usage scenarios will be considered here: the manipulation of a single sound and cross-synthesis.

2.1 Single-sound manipulation

Fig.2 shows the Factorsynth interface in a single-sound manipulation scenario, running on Max 7. Note that the

[1] `http://www.jjburred.com/software/factorsynth`

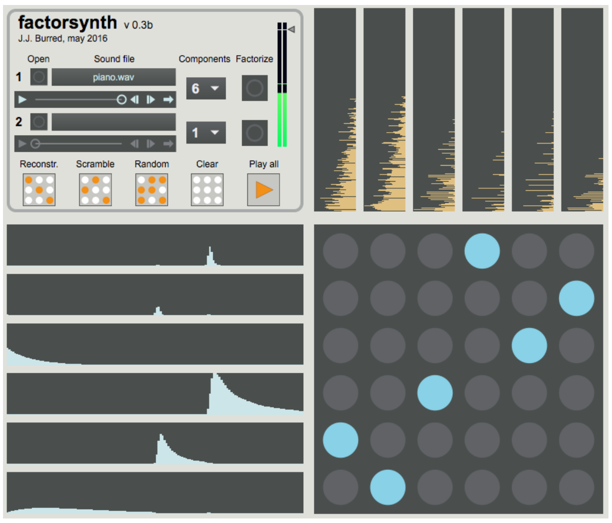


Figure 2. Main interface of Factorsynth for processing of a single sound.

interface allows to load two files; for single-sound processing, only the first file is loaded. Next to the file name is a menu to select the number of components for the extraction. Remember that one component corresponds to one base/activation pair. In Fig.2, six components have been set for factorization.

When the number of components is chosen, the size of the switchboard and the number of display areas for the bases and activations are adjusted. NMF decomposition is launched when clicking on the ‘factorize’ button corresponding to the loaded sound. Computation time is around 25% of the length of the input file (a 4s file will take 1s to decompose). After computation, the display areas are filled with the bases (above) and the activations (at the left). The bases are displayed in logarithmic amplitude and linear frequency.

Clicking the ‘factorize’ button again repeats factorization. Successive factorization runs can produce slightly different results since NMF is a numerical optimization algorithm that relies on random initialization [2]. This can result in small amplitude differences and, more noticeably, a different ordering of the output bases and activations.

There is a scale ambiguity of the factors produced by any factorization, since a product $(cx) \times (y/c)$ is the same for any value of c. In other words, it is possible to arbitrarily transfer energy from the bases to the activations, or vice-versa, without changing the validity of the factorization. In Factorsynth, the following convention has been applied:

1. First, the spectral bases are *individually* max-normalized (i.e., re-scaled so that they all reach the maximum amplitude of the display area) and the resulting energy differences transferred to the activations.

2. Then, for display, the activations are *globally* max-normalized (i.e., re-scaled so that only one reaches the maximum amplitude).

[2] The interested reader can find details about the NMF algorithm in the extensive literature (e.g.: [7]).

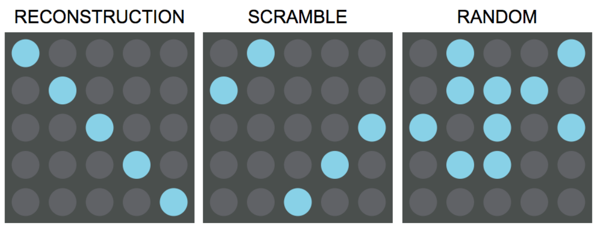


Figure 3. Modes for automatic switchboard configuration.

What this means in terms of interpretation of the graphical output is that the energy information is contained in the activations. A low-energy component will have a low-amplitude activation (such as the last component at the bottom of Fig.2), but its spectral base will still span the whole display range.

The bases and activations are displayed on editable multi-slider objects, so that the user can draw on them to modify the sound to be resynthesized.

The user can then click on the switchboard buttons to assign the desired base/activation pairs for the resynthesis. When clicking on a switchboard button, a resynthesis and playback of the corresponding base/activation pair is instantly launched (the computation time needed for resynthesis is negligible) in order to listen to that separate component. Thus, buttons on the diagonal will play coupled base/activation pairs which were present in the original sound, and off-diagonal buttons will generate artificial components not originally present. Once the desired individual connections have been made, the full resynthesized sound can be played by clicking on the 'play all' button.

Instead of manually selecting the switchboard connections, there are 4 buttons to set them up automatically (see Fig.3):

- **Reconstruction**. Sets the diagonal buttons on, all the others off. When full resynthesis is performed, this results in the playback of an approximation of the original sound. Reconstruction is never identical to the input, since NMF, like most factorization algorithms, is approximate.

- **Scramble**. Generates a random permutation of the connections. The connections are one-to-one (injective).

- **Random**. All connections are randomly chosen. Repetitions are possible: a single activation can control several bases, or several activations can control a single base.

- **Clear**. Sets all connections to zero.

2.2 Cross-synthesis

When two input sound files are selected, the interface enters in cross-synthesis mode (Fig.4). The switchboard is divided into four parts. The two parts on the diagonal (with the blue buttons) correspond to the single-sound manipulation connections, controlling the base/activation combinations within each of the input sounds. The two other

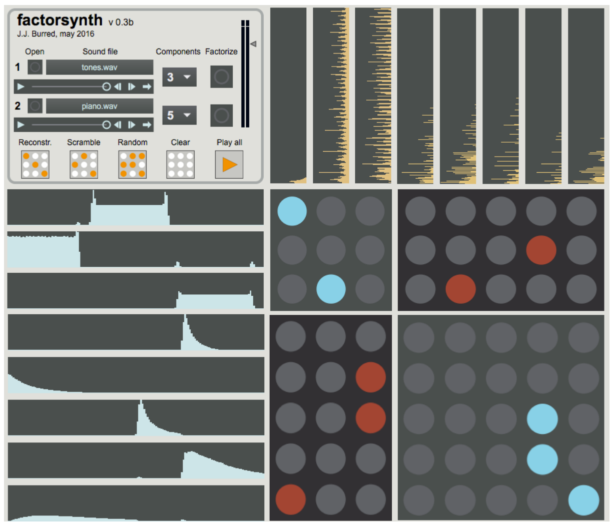


Figure 4. Factorsynth interface in cross-synthesis mode.

sectors of the switchboard, distinguished by their red buttons, control connections between bases of one sound and activations of the other, thus generating *cross-components*.

In this type of factorization-based cross-synthesis [5], internal temporal elements of one sound can thus control internal spectral shapes of the other.

3. RESYNTHESIS

It is worth going into some detail about the resynthesis process in order to understand the sonic results of Factorsynth. As its name implies, NMF works only on real, non-negative numbers, which means that phase information is discarded and only magnitude or power spectrograms are taken as the input. The combination of bases and activations (also comprised of real numbers) produce a set of magnitude spectrograms from which the synthesized output sounds have to be generated. Since the phase information was discarded from the outset, there are two options at this point:

- Either new phase information is generated randomly or by means of an optimization method, such as the Griffin and Lim algorithm [8], or

- Phase information is taken from the original input complex spectrogram.

The second option has been chosen for Factorsynth due to its superior sound quality and faster computation time. However, instead of directly attaching the input phases to the output spectrogram, Factorsynth uses Wiener filtering [6], which is known from source separation to produce more natural sounds.

Wiener filtering consists of computing a time-frequency mask from the output magnitude spectrograms that, when applied (by element-wise multiplication) to the input complex spectrogram produces the output spectrogram. Such a Wiener mask can be understood as a time-varying filter that is, in effect, performing subtractive synthesis from the input sound. Once the output complex spectrogram has been obtained in this way, an overlap-add algorithm is applied to invert it and produce the output time-domain signal.

The choice of Wiener filtering for resynthesis has an important implication for Factorsynth: if a high-energy activation is combined with an originally unrelated basis, it can happen that the resulting component will nevertheless be of low energy. Indeed, frequency contents can be hardly amplified if there is only little energy at those frequencies in the corresponding position of the input sound.

Factorsynth is able to handle both mono and stereo signals. For stereo signals, NMF is applied to the sum of both channels, and the resulting time-frequency masks are applied to both left and right input spectrograms to generate each of the output channels.

4. THE FACTORSYNTH∼ EXTERNAL

The core of the Factorsynth Max patch is the factorsynth∼ external object. It implements both NMF decomposition and Wiener resynthesis. Each factorsynth∼ object handles a single input file, so for cross-synthesis, two instances are needed. Linear algebra operations inside the object (FFTs, matrix multiplications, outer products...) are implemented using Apple's highly optimized vDSP library, part of the Accelerate framework.

A usage example for both factorization and resynthesis is shown in Fig.5.

4.1 Factorization operation

The sequence of operations needed to perform a factorization is the following:

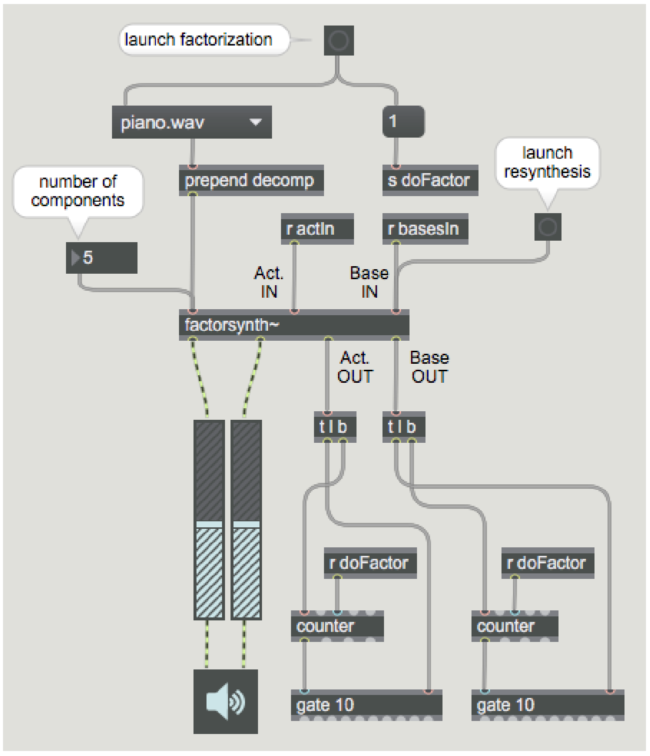


Figure 5. Usage example of the `factorsynth∼` external.

1. The number of components K is passed as an integer to the left inlet.

2. A message of the form `decomp filename` is passed to the left inlet, launching spectrogram computation and NMF factorization. The specific NMF algorithm implemented in factorsynth∼ is Kullback-Leibler (KL) NMF.

3. Activations are output from the third outlet as a sequence of K lists.

4. Bases are output from the fourth outlet as a sequence of K lists.

The output list sequences could then, for instance, be handled by gate objects to be sent to separate display areas, as shown in Fig.5.

4.2 Resynthesis operation

To launch a resynthesis with a given set of base/activations pairs, the following operation sequence must be performed:

1. A sequence of lists are read into the middle inlet, containing the activations.

2. A sequence of lists are read into the right inlet, containing the bases.

3. A bang is sent to the right inlet, signaling the end of the incoming data and launching the computation of the global Wiener mask and its application to the input spectrogram. There is no need to reload the input audio file since the spectrograms are stored in memory for every instance of a factorsynth∼ object.

4. The masked spectrogram is converted back into the time domain using the overlap-add technique. The resulting audio is output as a mono signal from the first outlet, or as a stereo signal from the left and second outlets.

5. FUTURE DEVELOPMENTS

Aside from being computationally intensive (as mentioned before, around 25% of the input file length), NMF factorization is an intrinsically off-line operation, since the full length of the input signal has to be observed prior to starting the decomposition algorithm. Thus, the current version of Factorsynth is non-real-time and works only on sound files. An important goal for future versions is the ability to process incoming audio data in real- or near-real-time.

A relatively straightforward way of implementing a real-time cross-synthesis would be to perform a preliminary factorization of a sound and then use an arbitrary selection of its stored spectral bases to filter the incoming audio stream. A second, most sophisticated way would be to explore online factorization algorithms [9] and assess the feasibility of a quick decomposition of the input stream.

Another direction for future developments will be the exploration of alternative interfaces for the representation and recombination of the extracted bases and activations. The current interface, based on displaying individual bases and activations, might become ineffective when using a large number of components (in source separation, tens of components are often used). In that case, an interface based on a 2-D scatter plot might be more appropriate, in which bases or activations could be represented as points and placed in coordinates corresponding to a given spectral or temporal shape feature. Connections in the cross-synthesis switchboard could then be generated automatically following criteria of proximity in such a feature space.

6. CONCLUSIONS

This paper has introduced Factorsynth, a graphical tool for the Max environment that exploits matrix factorization techniques to perform sound manipulations. Stemming from data analysis and machine learning, matrix factorization techniques remain relatively unknown in the field of computer music. It has recently been shown that such techniques constitute a promising new alternative to sinusoidal or source/filter models for analysis/resynthesis applications, and they allow a new kind of cross-synthesis that operates at the level of internal elements of the involved sounds (spectral shapes, salient temporal events...), rather than on global features. Factorsynth aims at bringing those new concepts to a wider audience of composers and sound designers. Its simple graphical interface visualizes all extracted elements and allows the user to modify them and carefully control their combination before resynthesis.

Acknowledgments

The author would like to thank Marco Liuni, Emanuele Palumbo, Carmine Emanuele Cella and Nicolas Obin for the insightful discussions and helpful suggestions.

7. REFERENCES

[1] D. D. Lee and H. S. Seung, “Learning the parts of objects by non-negative matrix factorization,” *Nature*, vol. 401, pp. 788–791, 1999.

[2] S. S. Topel and M. A. Casey, “Elementary sources: Latent component analysis for music composition,” in *Proc. ISMIR*, Miami, USA, 2011.

[3] R. Sarver and A. Klapuri, “Application of non-negative matrix factorization to signal-adaptive audio effects.” in *Proc. DAFX*, Paris, France, 2011.

[4] R. Maguire, “Creating musical structure from the temporal dynamics of soundscapes,” in *Proc. Int. Conf. on Information Sciences, Signal Processing and Applications (ISSPA)*, Montreal, Canada, 2012.

[5] J. J. Burred, “Cross-synthesis based on spectrogram factorization,” in *Proc. ICMC*, Perth, Australia, 2013.

[6] ——, “A framework for music analysis/resynthsis based on matrix factorization,” in *Proc. ICMC*, Athens, Greece, 2014.

[7] D. Lee and H. Seung, “Algorithms for non-negative matrix factorization,” in *Neural Information Processing Systems*, Denver, USA, 2001.

[8] N. Sturmel and L. Daudet, “Signal reconstruction from STFT magnitude: a state of the art.” in *Proc. DAFX*, Paris, France, 2011.

[9] A. Lefévre, F. Bach, and C. Févotte, “Online algorithms for nonnegative matrix factorization with the Itakura-Saito divergence,” in *Proc. WASPAA*, New Paltz, USA, 2011.