

SOUND DESIGN LEARNING FOR FREQUENCY MODULATION SYNTHESIS PARAMETERS

Juan-Pablo Cáceres

Center for Computer Research in Music and Acoustics (CCRMA)

Stanford University

j.caceres@ccrma.stanford.edu

1. OVERVIEW

Sound design is an extremely slow processes that requires a significant amount of experience to achieve specific goals. Through a process of trial and error the designer uses an existing synthesis engine or program its own to create a sound that is closer to a specified target. The sound target can be either an existing sound or a verbal specification.

Frequency Modulation (FM) Synthesis [1] uses a very efficient algorithm to match a wide variety of sounds. This technique has the advantage that through the use of a small number of control parameters, it can achieve a wide range of timbers. This flexibility makes it also hard to get to the “correct” parameters for a specific sound target.

A general heuristic to match synthesis parameters of a fixed sound engine to an arbitrary sound target is proposed. After the generation of training data from the synthesis engine, PCA is perform to extract relevant metrics and a combination of k-means and gradient descent is used to get an estimation.

2. PREVIOUS WORK

Andrew Horner [3] has applied genetic algorithms to match FM synthesis parameters. The engine topology can also vary with the added complexity of model selection. Genetic algorithms used lead to good results but are computationally expensive and slow, making them not suitable for real-time purposes.

A more general approach has been presented by Matt Hoffman [2], where simulated annealing is used as an alternative to genetic algorithms to match synthesis parameters. These algorithms are also very slow, so a database is filled and a general locality-sensitive hashing is used to get fast approximate lookups.

In general the problem with models with variable synthesis engine is that the more complex the more accurately the target can be matched. This naturally leads to spectral modeling synthesis [4], a generic approach to match any sound through spectral features. This is very accurate but very slow and the mapping problem of how to control the synthesis parameters is an open question.

3. PROPOSED FRAMEWORK

This work proposed a general approach to match parameters to a fixed synthesis engine. A fixed topology is preferred over an evolutionary one because:

- training and database generation based on the topology and parameters range is possible.
- a better control can be achieve in real-time performances without the need of a higher level mapping.

The steps consist on the following stages that will be discussed in details in what follows:

1. Define the synthesis engine and range of control parameters
2. Generate random training data spanning the space of the control parameters
3. Chose a feature vector and a distance metric and cost function to evaluate similarity
4. Perform iterative k-means clustering on the training set to generate a decision tree based on feature similarity
5. For a target sound, do a tree search on the tree database and use that as a starting point to perform gradient descent to reach the local minimum of the cost function

3.1. Synthesis engine

Synthesis engines like the ones available on commercial synthesizers or ad hoc computer synthesis graphs can be used, e.g., FM synthesis and subtractive synthesis. The control parameters are completely specified within a range.

3.2. Training data

To generate training data we simply generate random control parameters values (each of them within its specific range) so that we span a wide variety in the control space. Some algorithms like FM synthesis have strong nonlinearities, so in general what we want is to maximize the variety of features. To do that we do PCA and clustering on the feature space and not on the control space.

3.3. Feature vector and a distance metric

To match parameters for steady-state sounds, spectral feature are much more relevant than time features. In general, any spectral feature may be use that provides a good representation of the spectral characteristic of the family of sounds generated by the specified synthesis engine. PCA is then performed on the chosen feature vector to reduce dimensionality in the evaluation of similarity.

It has been showed that the MFCC (Mel frequency cepstral coefficient) is a perceptually valid metric for timbre [6]. It has also the advantage that it incorporates PCA on its arrangement, so performing PCA again won't be necessary. It is also more important to inform the system with psychoacoustic features since ultimately the target sound has to be similar psychoacoustically and not physically (although in many situation both coincide).

The MFCC coefficients are a representation of the spectrum into total energy per critical band using auditory filter banks. We use 13 coefficients that are computed with the Auditory Toolbox [5] in Matlab.

For the purpose of matching steady-state sounds, we define the sample mean MFCC vector \bar{c} as the average of the the successive MFCC across the chosen total time frame. The comparison between sounds is done in terms of the norm of the difference between the target feature vector and the estimated one.

$$D_{\bar{c}} = \bar{c}_{target} - \bar{c}_{estimated}$$

Independently of the metric chose (others may be more appropriate for other problems) the purpose is to minimize $D_{\bar{c}}$.

3.4. K-means clustering decision tree

We want to maximize the variability on the feature vector that represents the variability of the target sounds. In the training data we perform k-means clustering on the feature vectors. The Spider [7] is used to do the computations in Matlab. This is going to group the data into clusters sound features.

The clustering is done in an iterative way, in order to generate a decision tree for a fast lookup of the closest match of a target sound with the sounds on the database. The system defines a number of clusters N_c and a number of levels N_l , so we can span a total of $N_c^{N_l}$ plus the last level training data. For example, with 10 clusters and 3 levels, we can span a total of approximately 10^3 training examples

3.5. Gradient descent

After finding the closest match in our training database tree, we use that point as a starting point and perform gradient descent to get to the local minimum. Hopefully we are in a region where the local minimum is also the global one. The bigger the training data, the highest the chances that we reach the global minima.

4. FM SYNTHESIS EXAMPLE

A very simple frequency modulation engine is used as an example. The engine consists of two sinusoids, one acting as carrier and the other at modulator:

$$x(t) = A \sin(2\pi f_c t + I \sin(2\pi f_m t + \phi_m) + \phi_c)$$

f_c , f_m , I are used as control parameters. f_c , f_m have a range of 20 Hz to 1000 Hz and I goes from 0 to 10. The other parameters are fixed at $A = 1$, $\phi_m = 0$ and $\phi_c = 0$. To generate the training data, we use 128 discrete steps on the three control parameters and generate random combinations of them.

30000 random combinations are generated. This number is chosen to be 30 times bigger than the minimum $N_c^{N_l} = 10^3$, in order to avoid having clusters without data.

The clustering is done using 10 clusters and 3 levels.

4.1. K-means clustering

Figure 1 shows the first-level clusters of the training data. The cluster centroids are also plotted. It is clear that the clustering is grouping sounds with similar MFCC features. This clustering is done again in 3 levels, with 10 cluster groups each.

Figure 1. K-means clustering and cluster center for the MFCC of the FM example

4.2. System performance

The system performance was evaluated with the generation of random parameters to generate a target sound, and then try to match the parameters. Informal listening tests suggests that estimation is very accurate in terms of sound quality, but sometimes is in a slightly different pitch. This suggest that for a future work harmonicity coefficients and spectral centroid may be helpful in the feature vector to make the estimated parameters more accurate.

Table 1 shows for example of target and estimated parameters. The corresponding spectrograms are shown in Figures 2, 3, 4, 5.

We see that even when one of the parameters is far from the global minimum (Table 1) the spectrograms are still similar. This suggest that different configurations produce different sounds, and even if we don't get to the exact parameters of the original sounds, perceptually it is still acceptable.

All these examples need no more than 80 iterations, what makes the algorithms very fast, with less than a second to estimate the parameters in Matlab. The order of the iterations needed for genetic algorithms and simulated annealing is much higher.

Table 1. Center frequency FM parameter for each band.

	f_c	f_m	I
Target 1	440	440	5
Estimated 1	422.3	435.2	5.0373
Target 2	100	30	3
Estimated 2	96.1	25.5	4.0488
Target 3	800	800	9
Estimated 3	831.1	806.1	2.1818
Target 4	900	100	7
Estimated 4	592.3	102.2	9.7604

Figure 3. Spectrogram comparison for example 2

Figure 4. Spectrogram comparison for example 3

Figure 2. Spectrogram comparison for example 1

5. CONCLUSIONS AND FUTURE WORK

A fast algorithm was implemented for a simple example of FM synthesis. The example performs very fast and does not require a huge database to store the training data. However we need to review the feature vector in order to make it more accurate. Different distance metrics that can be tried are the raw spectrogram with reduced dimensionality, and the addition of harmonicity and spectral centroid to the MFCC coefficients.

Figure 5. Spectrogram comparison for example 4

In some synthesis engines the use of Kernel PCA may be useful to account for the non-linearities of the system. A similar approach of the k-means tree implemented may be tried using kd-tree algorithm.

Other extensions include implement more complex systems, like second order FM synthesis and use a localized global minimum algorithm to find the global minimum in the region given by the k-means tree estimation but without getting stuck on a local minimum for systems with strong non-linearities.

Testing the system with sounds not generated with the system is also planned.

6. ACKNOWLEDGMENTS

I would like to thanks Hiroko Terasawa for the advice on psychoacoustic metrics and Stefano Corazza for his valuable ML algorithms sugestions.

References

- [1] J. Chowning. The synthesis of complex audio spectra by means of frequency modulation. *Journal of the Audio Engineering Society*, 21(7):526–534, 1973.
- [2] M. Hoffman and P. R. Cook. Real-time feature-based synthesis for live musical performance. In *NIME '07: Proceedings of the 7th international conference on New interfaces for musical expression*, pages 309–312, New York, NY, USA, 2007. ACM.
- [3] A. Horner, J. Beauchamp, and L. Haken. Machine tongues XVI: Genetic algorithms and their application to FM matching synthesis. *Computer Music Journal*, 17(4):17–29, Winter 1993.
- [4] X. Serra and J. O. Smith. Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition. *Computer Music Journal*, 14(4):12–24, Winter 1990.
- [5] M. Slaney. Auditory toolbox, version 2. Technical Report 1998-10, Interval Research Corporation, Palo Alto, California, USA, 1998.
- [6] H. Terasawa, J. Berger, and J. O. Smith. Using a perceptually based timbre metric for parameter control estimation in physical modeling synthesis. In *Proceedings of International Computer Music Conference*, 2005.
- [7] J. Weston, A. Elisseeff, G. Baklr, and F. Sinz. The Spider.