MORPHING OF GRANULAR SOUNDS

Sadjad Siddiq
Advanced Technology Division,
Square Enix Co., Ltd.
Tokyo, Japan
siddsadj@square-enix.com

ABSTRACT

Granular sounds are commonly used in video games but the conventional approach of using recorded samples does not allow sound designers to modify these sounds. In this paper we present a technique to synthesize granular sound whose tone color lies at an arbitrary point between two given granular sound samples. We first extract grains and noise profiles from the recordings, morph between them and finally synthesize sound using the morphed data. During sound synthesis a number of parameters, such as the number of grains per second or the loudness distribution of the grains, can be altered to vary the sound. The proposed method does not only allow to create new sounds in real-time, it also drastically reduces the memory footprint of granular sounds by reducing a long recording to a few hundred grains of a few milliseconds length each.

1. INTRODUCTION

1.1. Granular synthesis in video games

In previous work we described the morphing between simple impact sounds [1]. In this paper we focus on morphing between complex sounds that consist of a large amount of similar auditory events. Examples of such sounds are the sound of rain, consisting of the sound of thousands of rain drops hitting the ground; the sound of running water, which is essentially the sound of thousands of resonating air bubbles in water [2]; the sound of popping fireworks or the sound of breaking rock [3]. Since we use granular synthesis to synthesise such sounds in real-time we call these sounds "granular sounds" and the auditory events they consist of "grains".

Although such sounds are used extensively in video games, they are rarely produced by granular synthesis. The most common approach is to rely on recorded samples, but this does not give sound designers much control over the sound. Additionally such samples are usually heavy on memory, because they must be quite long to avoid repetitiveness. Using granular synthesis to create such sounds from prerecorded or synthesized grains is a method that is much lighter on memory and gives sound designers a lot of freedom to modify the sound in multiple ways by modifying parameters of the synthesis [3].

Figure 1 summarizes the implementation of a simple granular synthesizer. In granular synthesis signals are produced by mixing grains. These grains are usually very short, in the range of 2 to 20 ms. The number of grains per second can be used as a parameter to control the granularity of the sound. A low number would allow the listener to perceive individual grains while a high number would result in a large amount of grains overlapping each other and create a signal close to white (or colored) noise. Before mixing, grains are usually modified in a number of ways. In the simplest case their amplitude is attenuated randomly. Depending on the probability distribution of the attenuation factors, this can have a very drastic effect on the granularity and tone color of the produced sound. For a more detailed introduction to granular synthesis refer to [4] or [5].

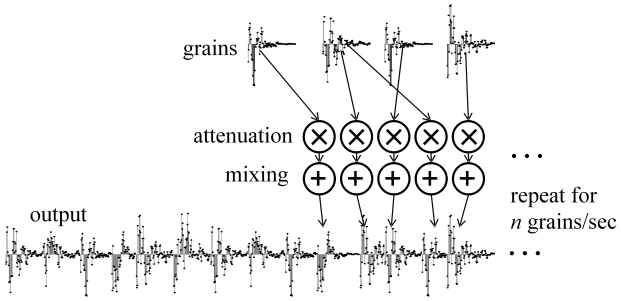

Figure 1: *Summary of granular synthesis. Grains are attenuated and randomly added to the final mix.*

When using granular synthesis in video games, the synthesis algorithm can be coupled to the physics engine to create realistic sounds. To implement the sound of breaking rock, as introduced in [3], we calculate the number of grains per second in the final mix based on the number of collisions between rock fragments as calculated by the physics engine. Also the distribution of the attenuation factors applied to the grains in the final mix is calculated based on the magnitude of impacts in the physics engine. The sounds of rain or running water can also be synthesized and controlled in a very flexible way when using granular synthesis.

1.2. About the proposed system

One reason why granular synthesis does not find widespread use in video games might be the necessity to record or synthesize appropriate grains. Recording can be problematic if the underlying single sound events are very quiet, like small rain droplets hitting the ground, or if they are hard to isolate, like resonating air bubbles in water. Synthesis can solve this problem in some cases, but deriving synthesis models to create appropriate grain sounds can be a time consuming process. In addition such models are often very hard to parametrize and control due to their level of abstraction.

In the system described in this paper grains are extracted automatically from short recorded samples of granular sounds to avoid the aforementioned issues. Using these grains, sounds that are very close to the original samples can be reproduced. To furnish sound designers with a straight-forward tool enabling them to design a variety of granular sounds, we combine automatic grain extrac-

tion with sound morphing. By morphing between the grains that have been extracted from two (or more) different recorded samples, sounds that lie perceptually at an arbitrary position between these samples can be created. Although this approach has the limitation that sound samples are needed to create sound - as opposed to a system where grains are synthesized - it has the big advantage that sound designers do not have to record grain sounds or deal with complicated grain synthesis models. Instead they can use existing recordings of granular sounds as a starting point for sound synthesis.

1.3. Related work

Several papers discuss automatic extraction of grains or other features from granular sounds.

Bascou and Pottier [6] automatically detect the distribution and pitch variation of grains in granular sounds. However, the grains they are working with are extracted manually. Their algorithm works by detecting pitch-shifted versions of these grains in the spectrogram of a signal.

Lee et al. [7] extract grains from audio signals based on sudden changes in the energy content of the spectrum, expressed as spectral flux between neighbouring frames. These grains are then used to resynthesize a signal in which the distribution of the extracted grains over time can be modified. To fill gaps between grains that are caused by a different distribution, grains are made longer by using linear prediction. Gaps can also be filled by concatenating similar grains. In their study grains do not overlap.

Schwarz et al. [8] extract grains from audio and arrange them in multidimensional space according to several features describing their tone color. Grains are extracted by splitting the input signal at parts that are silent or based on high-frequency content of the signal. The software tool CataRT [9], developed by the authors, allows users to generate sound by navigation through the tone color space, moving between groups of similar grains. This comes close to morphing between sounds, but requires users to provide appropriate grains for all needed tone colors.

Lu et al. [10] implement sound texture synthesis using granular synthesis. They extract grains from an input signal based on MFCC-similarity measurement between short frames. Grains are varied in various ways during resynthesis to create variations of the sound.

The extracted grains used by Lu et al. [10] are between 0.3 s and 1.2 s long and thus much longer than the grains used in our method. Also Fröjd and Horner [11] cut the input signal into grains ("blocks") with a comparatively large optimal length of 2 s. The advantage of long grains as used in these studies is that micro structures of the sound can be captured in the grains. However, such microstructures can not be changed during resynthesis.

1.4. Structure of this paper

The next section describes the technique used for automatic grain extraction. Section 3 gives an overview on how sound can be synthesized from the extracted grains. The method used for morphing between granular sounds is described in section 4. Results of synthesis and morphing are reported in the last section. A link to sound samples is provided.

2. AUTOMATIC GRAIN EXTRACTION

2.1. Noise subtraction

In granular sounds that consist of a very dense pattern of grains, most of these converge to noise and only the louder ones can be perceived separately. To improve the quality of the extracted grains, we first remove this noise by spectral subtraction (see [12] for a detailed introduction).

To determine the spectrum of the noise to be removed, the sample is first cut into overlapping frames. We use frames of 1024 samples that are extracted from the signal at increments of 32 samples. The reason for choosing a small increment is to maximize the number of extracted frames, since the loudest 85 % of all frames are rejected later. This was found to be a simple measure to reduce the number of frames containing loud and short bursts, which are typically found in granular sounds. A small overlap is also needed to ensure quality in the resynthesized signal after noise subtraction. After applying a Hamming window each frame is transformed to the frequency domain using a FFT of size N_{fft}, which is equal to the frame length. Then the amplitude spectrum of each frame is calculated. The spectra are smoothed by applying a FIR filter whose impulse response is a Gaussian calculated by the equation

$$h[n] = a \cdot e^{-\frac{(n-b)^2}{2c^2}}, \tag{1}$$

where $n = [0, 1, ..., 33]$, a is a normalization constant so that $\sum h[n] = 1$, $b = 16$ and $c = 3$.

The energy $e[m]$ of each frame m is calculated as

$$e[m] = \sum_{n=0}^{N_{\text{fft}}/2+1} S_a[m, n]^2, \tag{2}$$

where $S_a[m, n]$ is the amplitude of the frequency bin n in the smoothed spectrum of frame m.

To avoid loud sound events that distort the extracted noise spectrum, only the quietest 15 % of all frames are used to calculate the noise spectrum S_n, which is calculated as the average amplitude spectrum of these frames.

To reconstruct the signal with reduced noise, the noise spectrum S_n is first subtracted from the amplitude spectra S_a of each frame, setting all bins to zero where the amplitude of the noise spectrum is higher:

$$S'_a[m, n] = \begin{cases} S_a[m, n] - S_n[n], & \text{if } S_a[m, n] > S_n[n] \\ 0, & \text{if } S_a[m, n] \leq S_n[n] \end{cases} \tag{3}$$

Then the amplitude spectra with reduced noise S'_a are applied to the complex FFT coefficients $S[m, n]$ of each frame after normalizing them.

$$S_c[m, n] = S'_a[m, n] \frac{S[m, n]}{S_a[m, n]}, \text{where } n = 0, 1, ..., N_{\text{fft}}/2 \tag{4}$$

The second half of the FFT coefficients, i.e. the frequency bins $n = N_{\text{fft}}/2 + 1, ..., N - 1$ of all frames are obtained by mirroring the complex conjugate:

$$S_c[m, n] = S_c^*[m, N_{\text{fft}} - n], \text{where } n = \frac{N_{\text{fft}}}{2} + 1, ..., N_{\text{fft}} - 1 \tag{5}$$

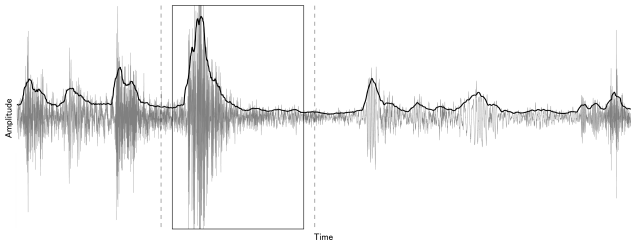

Figure 2: *Extraction of a grain from granular sound. The extracted grain (solid rectangle) is found between the boundaries of the window (dotted lines) placed around the maximum of the envelope (solid curve).*

The resulting complex spectra are then transformed to the time domain using the inverse FFT and overlapped according to the way the frames where extracted initially, which yields the signal with reduced noise $s_c[t]$.

The noise spectrum S_n is also used when resynthesizing the sound and when morphing between different sounds.

2.2. Grain extraction

The grains are extracted from the loudest parts of the signal $s_c[t]$. The loudest parts are found by looking at its smoothed envelope $g[t]$. The envelope $g'[t]$ is obtained using $\hat{s}_c[t]$, the Hilbert transform of the signal and its complex conjugate $\hat{s}_c^*[t]$:

$$g'[t] = \sqrt{\hat{s}_c[t] \cdot \hat{s}_c^*[t]} \tag{6}$$

The smoothed envelope $g[t]$ is obtained by applying a moving average of length 100.

As is shown in figure 2, the first grain is extracted from a window around the index τ, which is the position of the global maximum of the envelope. The index t_s of the first sample to be extracted and the index t_e of the last sample are determined by finding the minima of the envelope that lie within a certain window around τ. The window is defined by the variables w_s and w_e, which denote the maximum number of samples between τ and the start or the end of the window respectively. The grain is thus extracted between

$$t_s = \min(g[t] : \tau - w_s \leq t < \tau) \text{ and} \tag{7}$$

$$t_e = \min(g[t] : \tau < t \leq \tau + w_e). \tag{8}$$

The grain is stored in the grain waveform table but deleted from the signal $s_c[t]$ and the envelope $env[t]$ by setting both in the range $[t_s, t_e]$ to zero. After this deletion the same algorithm is reiterated to find the next grain. This is repeated until the user specified number of grains have been extracted or until the envelope is all zero.

To reduce unwanted noise during later synthesis, the start and end of all extracted grains are attenuated gradually so that jumps in the final mix are avoided. This is done by attenuating with the envelope a, which is defined as

$$a[t'] = \begin{cases} \sqrt[4]{\frac{t'}{\tau - t_s}}, & \text{for } t_s \leq t < \tau \\ \sqrt[4]{1 - \frac{t-\tau}{t_e - \tau}}, & \text{for } \tau \leq t \leq t_e \end{cases} \tag{9}$$

where $t' = t - t_s$.

3. SOUND SYNTHESIS USING EXTRACTED GRAINS

To synthesize sound of arbitrary length that is similar to the original sound from which noise and grains were extracted, we first synthesize noise according to the noise spectrum S_n, which was extracted as described in section 2.1. Then we add the extracted grains to the noise signal.

The noise is generated by shaping white noise according to the extracted noise spectrum S_n using multiplication of the spectra in the frequency domain. After generating a block of white noise of length $N_{\text{fft}}/2$, i.e. half the length of the FFT used when creating the noise spectrum to avoid aliasing, it is padded with zeros to form a block of N_{fft} samples and transformed to the frequency domain using the FFT, yielding the frequency domain white noise signal $R'[n]$. Since the time domain signal was real valued, $R'[n]$ is symmetric and the multiplication only has to be applied to the first half:

$$R[n] = S_n[n]R'[n], \text{where } 0 \leq n \leq N_{\text{fft}}/2 \tag{10}$$

The product of this multiplication is mirrored to form the second half of the frequency domain representation

$$R[n] = R^*[N_{\text{fft}} - n], \text{where } N_{\text{fft}}/2 < n < N_{\text{fft}} \tag{11}$$

The shaped noise is obtained by transforming $R[n]$ back to the time domain. To create longer noise signals, multiple white noise buffers of length $N_{\text{fft}}/2$ have to be processed in this way and overlapped with length $N_{\text{fft}}/2$ after transformation to the time domain.

After creating noise, the extracted grains are added. Adding the grains at the same positions from which they were extracted with their original amplitude will create a signal that is very close to the original - not only acoustically, but also with regard to the actual waveform. However, to synthesize a signal that sounds similar to the original the grains do not need to be distributed in the exact same way as they were in the original. Instead, grains are placed randomly in the synthesized sound.

Especially when working with few extracted grains, repetitiveness can be avoided when the amplitude of the grains in the synthesized signal is varied randomly. To create a sound that resembles the original this variation should follow the same loudness distribution as the grains in the original sound. This can be achieved by the following method: Before synthesis all grains are normalized, so that their amplitude is one, but their original amplitude is stored in a separate array. During synthesis each grain is attenuated with an amplitude that is randomly drawn from the array of amplitudes. Alternatively random numbers can be drawn from a probability distribution that matches the original grain amplitude loudness distribution. Depending on the nature of the sound big variations in the loudness of the grains might not be desirable, because the tone color of quiet and loud grains are different. In such cases small variation of the original amplitude can help to prevent repetitiveness in the synthesized sound.

However, since not all grains can be extracted during grain extraction, synthesis that is only based on the number of grains extracted per second and their loudness distribution can yield very unsatisfactory results. Modifying these parameters can increase the quality of the synthesized sound. This also gives sound designers more control over the produced sound. They can modify the sound by varying the loudness of the noise, the loudness distribution of

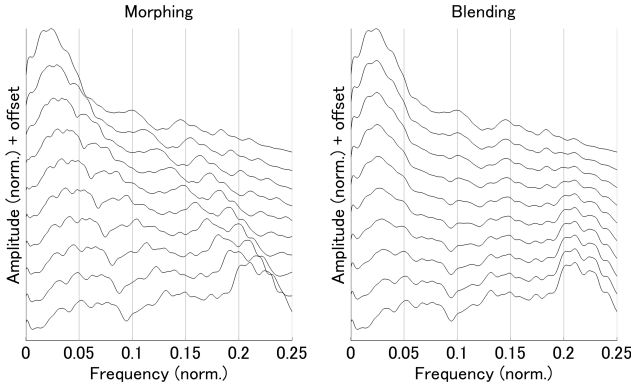

Figure 3: *Gradual morphing (left) and blending (right) between the two spectra at the top and bottom of the graphs. Note how formants move along the frequency axis when morphing.*

the grains and the number of grains per second. Finding appropriate values for these parameters is an important step in creating realistic sound.

Some granular sounds are the result of thousands of grains sounding at the same time. When synthesizing such sounds the noise component plays a significant role in recreating a realistic signal, but also grains should be layered on top of each other. The number of grains extracted per second only reflects the number of a few loud grains that were detected, but the number of grains actually sounding per second can be much higher. In such cases realism can be greatly increased by adding a higher number of grains to the final mix, attenuating grains with a loudness distribution that favours small amplitudes.

Section 5 presents some synthesis results along with the configurations used to create them.

4. MORPHING OF GRANULAR SOUNDS

4.1. Morphing vs. Blending

Noise and grains that are extracted from two different granular sounds can be combined in different ways. The straight-forward way of taking the (weighted) average of the noise spectra and using grains from both sounds will yield a sound that corresponds more to mixing both original sounds than to creating a sound whose tone color lies perceptually between the original sounds. This can be desirable when blending between one sound to the other sound.

However, to morph between the original sounds as described in section 1.2, i.e. to create a sound whose tone color lies between the original sounds, a different method must be used. One important manifestation of tone color is the shape of the spectrum of a sound because the distribution of energy over the frequency components of a sound plays an important part in tone color perception [13]. In the noise spectra in figure 3 the different energy distributions can clearly be seen. To gradually morph the tone color of one sound to the other sound it is necessary to move the formant(s) of one spectrum to the positions of the formant(s) in the other spectrum, also changing their shape gradually. Taking averages of both spectra with gradually changing weights, as shown in the right side of the figure, does not have this effect. Morphing between two spectra is shown in the left side of the figure.

The next sections describe the technique used for morphing and its application to granular sounds. This technique is applied to the noise and the grains extracted from the input sounds as described earlier.

4.2. Morphing of spectra

Shifting the formants of some spectrum A to the location of the formants of another spectrum B is essentially the same as redistributing the energy in spectrum A so that it resembles the energy distribution in spectrum B. The method used to achieve this was already introduced in an earlier publication [1], but for completeness we reproduce its explanation here.

We can express the energy distribution over the samples of an power spectrum by its integral. The integral of the power spectrum $s(\omega)$ where $\omega = [0; \Omega]$,

$$S(\omega) = \int_0^{\omega} s\left(\theta\right) d\theta, \tag{12}$$

expresses the energy between frequency ω and zero frequency. In other words, $S(\omega)$ is the cumulative energy of the spectrum.

We use the normalized cumulative energy of two spectra to match frequencies of the same cumulative energy and to find a spectrum with an intermediary energy distribution. To normalize we first remove any zero offset of the power spectrum

$$s_0(\omega) = s(\omega) - \min(s) \tag{13}$$

and then divide by the cumulative energy of s_0 at $\omega = \Omega$, which corresponds to the integral:

$$s_{\mathrm{norm}}\left(\omega\right) = s_0(\omega) / \int_0^{\Omega} s_0\left(\theta\right) d\theta \tag{14}$$

Then we calculate the integral of s_{norm}

$$S_{\mathrm{norm}}(\omega) = \int_0^{\omega} s_{\mathrm{norm}}\left(\theta\right) d\theta \tag{15}$$

whose maximum value $S_{\mathrm{norm}}(\Omega) = 1$ due to the normalization.

We do this for the two spectra $a(\omega)$ and $b(\omega)$ to get the normalized cumulative power spectra $A(\omega)$ and $B(\omega)$, keeping the normalization parameters

$$p_a = \min(a), \tag{16}$$

$$p_b = \min(b), \tag{17}$$

$$q_a = \int_0^{\Omega} a_0\left(\theta\right) d\theta \text{ and} \tag{18}$$

$$q_b = \int_0^{\Omega} b_0\left(\theta\right) d\theta. \tag{19}$$

We interpolate by first finding frequencies ω_a and ω_b in spectra A and B where the cumulative energy is equal to an arbitrary level y. In the interpolated spectrum, the frequency ω_{ab} where the cumulative energy reaches y should lie between ω_a and ω_b. We calculate this frequency as $\omega_{ab} = v \cdot \omega_a + (1 - v) \cdot \omega_b$ with v in

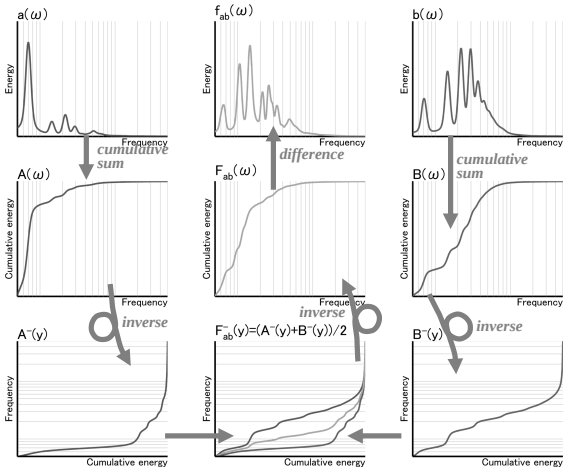

Figure 4: *Summary of the spectra morphing algorithm. Follow the arrows to read the figure. Given the two power spectra $a(\omega)$ and $b(\omega)$, their cumulative power spectra $A(\omega)$ and $B(\omega)$ are calculated. These are inverted to functions of cumulative energy, $A^-(y)$ and $B^-(y)$. Their average $X^-_{ab}(y)$ is inverted back to a function of frequency yielding the cumulative sum $X_{ab}(\omega)$, which is differentiated to get the morphed spectrum $x_{ab}(\omega)$.*

the range $[0; 1]$ expressing the desired similarity of the interpolated spectrum to A. This is the same as calculating the weighted average along the frequency axis between the cumulative energy curves of spectrum A and B. We therefore need to invert $A(\omega)$ and $B(\omega)$, which are functions of the frequency ω, to $A^-(y)$ and $B^-(y)$ which are functions of the cumulative energy y. We then calculate the weighted average of these inverses and get

$$F^-_{ab}(y) = v \cdot A^-(y) + (1 - v) \cdot B^-(y) \tag{20}$$

which is the inverse of the cumulative energy. Inverting and differentiating this function gives us the normalized interpolated power spectrum $f_{ab_{\mathrm{norm}}}$. To denormalize the spectrum we use the normalization parameters mentioned above and get

$$f_{ab} = (vq_a + (1 - v)q_b) \cdot X_{ab_{\mathrm{norm}}} + vp_a + (1 - v)p_b. \tag{21}$$

For implementation we need to consider discrete spectra. Supposing $f_{\mathrm{norm}}[n]$ is a discrete normalized power spectrum, the cumulative energy is calculated as the cumulative sum of its samples:

$$F_{\mathrm{norm}}[n] = \sum_{i=0}^{n} f_{\mathrm{norm}}[i] \tag{22}$$

To calculate its inverse we interpolate frequency values at arbitrary energy intervals. By varying the size of the interval we can control the quality of the output. After calculating the weighted average of the interpolated inversions we need to invert (interpolate) this average again to get the cumulative power spectrum as a discrete function of frequency. Calculating the difference between succeeding elements ($f[n] = F[n] - F[n-1]$) allows us to get the normalized interpolated power spectrum. After denormalizing in the same fashion as for continuous spectra, we get the interpolated power spectrum. The implementation is summarized in figure 4.

4.3. Morphing of noise spectra

The noise spectra of both sounds are morphed according to the algorithm described above. The resulting spectrum is used to shape the white noise as described in section 3.

4.4. Morphing of grains

4.4.1. Overview

The morphing of grains is not as straight forward as the morphing of the noise spectra: There are several grains for each sound, so before morphing it must be decided between which grains of sound A and which grains of sound B morphing should be conducted. Additionally grains are waveforms and cannot be represented by a single power spectrum, because temporal information of the signal would be lost.

The next paragraph describes how grains are first paired and then cut into frames and morphed.

4.4.2. Pairing grains

We want to morph between similar grains, so we need to find a way to measure the distance between two grains based on their similarity. This measurement is based on the spectral shape, which is one feature of the tone color. The distance is measured by the difference in the energy distribution between the frequency spectra of two grains. To calculate the difference between grains A and B, the frequency spectra are calculated over the whole length of both grains by first dividing the grains into overlapping frames of length $L = 256$ samples, extracted at increments of $d = 16$ samples to ensure a high time resolution, before transforming them to the frequency domain. The frames are padded with zeros to form blocks of length $N_{\mathrm{fft}} = 2L$ before applying the FFT. This extra space is needed to avoid aliasing caused by later multiplication in the frequency domain. Applying the FFT of length N_{fft} to frame m_{A} of grain A yields the complex FFT coefficients $X_{\mathrm{A}}[m_{\mathrm{A}}, n]$. From these coefficients power spectra are calculated for each frame using the formula

$$E_{\mathrm{A}}[m_{\mathrm{A}}, n] = |X_{\mathrm{A}}[m_{\mathrm{A}}, n]|^2, \text{ for } 0 \leq n \leq N_{\mathrm{fft}}/2, \tag{23}$$

where $E_{\mathrm{A}}[m_{\mathrm{A}}, n]$ are the resulting power spectra. The power spectrum $\overline{E_{\mathrm{A}}}[n]$ representing the whole grain is calculated by averaging all power spectra of the grain. The same calculations are executed for grain B.

The distance between both grains is calculated as the difference between the energy distribution of the power spectra $\overline{E_{\mathrm{A}}}[n]$ and $\overline{E_{\mathrm{B}}}[n]$. As in section 4.2, the energy distribution is expressed by the normalized cumulative sum of the power spectra

$$S_{\mathrm{A}}[n] = \sum_{i=0}^{n} \overline{E_{\mathrm{A}}}[i] / \sum \overline{E_{\mathrm{A}}}[i] \tag{24}$$

and $S_{\mathrm{B}}[n]$ which is calculated similarly. The difference in the energy distribution is measured by calculating the size of the area between the graphs of $S_{\mathrm{A}}[n]$ and $S_{\mathrm{B}}[n]$ as shown in figure 5.

Once the distances between all grains of sound A to all grains of sound B have been calculated, grains are paired using a variation of the stable marriage problem. Since the number of grains of both sounds is not necessarily equal, grains of the sound with more grains can be paired with more than one grain of the other

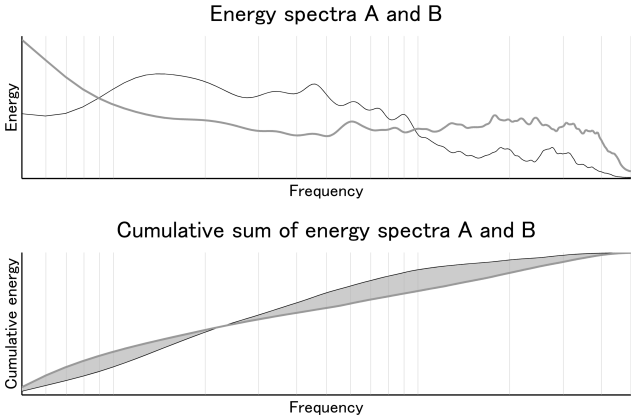

Figure 5: *The distance between the two spectra in the upper plot is measured by calculating the size of the area between the lines representing the cumulative sum of the spectra, as shown in the lower plot.*

sound. The following explanation uses the metaphor of companies (grains of the sound with more grains) and applicants (grains of the sound with less grains) to avoid venturing into the domain of polygamy. Two grains are considered to be a good match if their distance is small. Since the number of companies N_{co} and the number of applicants N_{ap} is not equal, every company has to hire $N = N_{\mathrm{co}}/N_{\mathrm{ap}}$ candidates on average so that all applicants are hired. The maximum number $N_{\max}$ of candidates a company can hire is fixed at the integer above N: $N_{\max} = \lceil N \rceil$.

Each iteration every applicant without a job offer applies at his most preferred company among the companies he has not yet applied at. After all applicants have filed applications, companies which have more than $N_{\max}$ applicants reject the worst matching applicants, keeping only $N_{\max}$ applications. These matches are tentative and can be rejected in the next iteration.

The globally worst matching applications of all companies are also rejected until the average number of applicants per company is equal to or lower than N. The algorithm is reiterated until all applicants are employed by a company.

Every grain in the applicant group thus has one match in the company group. Every grain in the company group has one or more matches in the applicant group, but is only paired with the best matching applicant.

4.4.3. Implementation of grain morphing

Once each grain of sound A is linked to a similar grain in sound B and vice-versa, morphing between two paired grains A and B can be considered.

As mentioned before, frames are extracted from the grains. Since the number of frames in grain A (M_{A}) and the number of frames in grain B (M_{B}) are not necessarily equal due to differing length of grains A and B, frames need to be aligned in a certain way. For the morphing of grains linear alignment yielded good results. The number of frames M_{AB} in the morphed sound is determined by $M_{\mathrm{AB}} = \mathrm{round}((1-v)M_{\mathrm{A}} + vM_{\mathrm{B}})$, where v is again the morphing factor (see section 4.2). The calculation of frame m of the morphed sound is based on frame m_{A} and m_{B} of sounds A and B respectively, which are chosen using the following formula:

$$m_{\mathrm{A}} = \mathrm{round}(M_{\mathrm{A}} \cdot m/M_{\mathrm{AB}}) \tag{25}$$

$$m_{\mathrm{B}} = \mathrm{round}(M_{\mathrm{B}} \cdot m/M_{\mathrm{AB}}) \tag{26}$$

Morphing is conducted between the power spectra $E_{\mathrm{A}}[m_{\mathrm{A}}, n]$ and $E_{\mathrm{B}}[m_{\mathrm{B}}, n]$ of the aligned frames m_{A} and m_{B}, which results in the morphed power spectrum E_{AB}. Since power spectra do not contain information about the phase of the signal, this information has to be extracted from the FFT coefficients $X_{\mathrm{A}}[m_{\mathrm{A}}, n]$ and $X_{\mathrm{B}}[m_{\mathrm{B}}, n]$ which were calculated earlier for frames m_{A} and m_{B} respectively. The phase is retained in the normalized complex spectra C_{A} and C_{B} which are calculated from the complex FFT coefficients. The formula to calculate the normalized complex spectrum C_{A} is

$$C_{\mathrm{A}}[m_{\mathrm{A}}, n] = \frac{S_{\mathrm{A}}[m_{\mathrm{A}}, n]}{|S_{\mathrm{A}}[m_{\mathrm{A}}, n]|}, \text{ where } 0 \leq n \leq N_{\mathrm{fft}}/2. \tag{27}$$

The spectrum C_{B} is calculated in the same way.

The morphed power spectrum E_{AB} is then applied to the complex spectra C_{A} and C_{B} of the aligned frames of both sounds to calculate the morphed spectrum S_{AB} of frame m, which is a weighted average of both sounds:

$$\begin{aligned} S_{\mathrm{AB}}[m, n] = (1-v) \cdot C_{\mathrm{A}}[m_{\mathrm{A}}, n] \cdot \sqrt{E_{\mathrm{AB}}[m, n]} \\ + v \cdot C_{\mathrm{B}}[m_{\mathrm{B}}, n] \cdot \sqrt{E_{\mathrm{AB}}[m, n]}, \end{aligned} \tag{28}$$

where $0 \leq n \leq N_{\mathrm{fft}}/2$.

The morphed spectrum is mirrored to obtain a complete set of FFT coefficients:

$$S_{\mathrm{AB}}[m, n] = S^*_{\mathrm{AB}}[m, N_{\mathrm{fft}} - n], \text{ where } N_{\mathrm{fft}}/2 < n < N_{\mathrm{fft}} \tag{29}$$

Then it is transformed to the time domain with the inverse FFT to obtain a time domain signal representation of the morphed frame. Finally, to form the final output of the morph, the frames are overlapped with spacing their start points at intervals of d samples, which is the same spacing used during frame extraction. When overlapping the frames a window function can be applied. A Hann window of length $N_{\mathrm{fft}}/2$ followed by $N_{\mathrm{fft}}/2$ zeros - by which the second half of the signal is discarded - yielded good results.

4.5. Real-time implementation

Although sound morphing can easily be implemented in real-time using the method described above (see [1] for further details), the high number of grains (around 200-500 per sound) make real-time implementation very difficult for synthesized granular sounds when grain morphing is conducted at run-time. This is why morphing between grains is executed before run-time instead. To enable users to create a sound with a tone color that lies at an arbitrary position between sound A and B, or - in other words - to create a sound corresponding to an arbitrary value of $v \in [0; 1]$ (as introduced in section 4.2), several sets of grain morphs for several values of v are prepared. The higher the number of grain morph sets, the higher is the smoothness of the morph between the two granular sounds. To implement a system having a resolution of ten sets, eight grain morphs with $v = 0.1; 0.2...0.9$ can be created and stored in memory in addition to the original grains of sounds A and B corresponding to $v = 0$ and $v = 1$ respectively. At runtime grains are chosen randomly from the sets closest to a given v value. The morphing of a single spectrum, however, is very cheap, so morphing of the noise can be done in real-time.

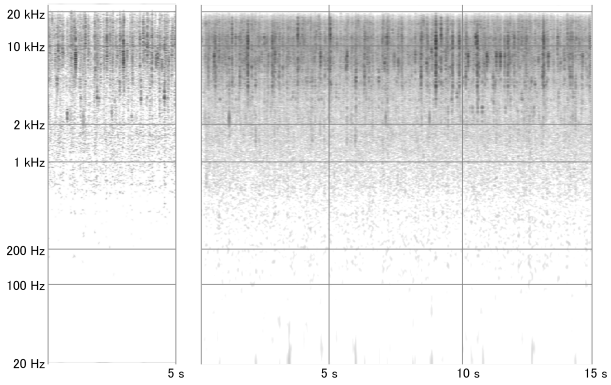

Figure 6: *Spectrogram of recording of rain sound (left) and synthesized rain sound (right).*

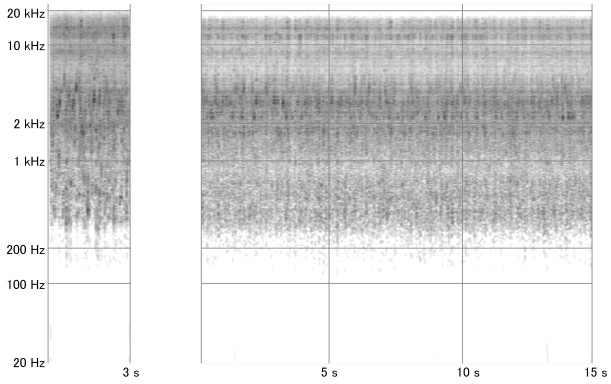

Figure 7: *Spectrogram of recording of water sound (left) and synthesized water sound (right).*

5. RESULTS

5.1. Synthesis

Realistic granular sounds that bear all the characteristics of the originals can be synthesized by extracting grains and noise from short samples of only a few seconds length. The sounds shown in figures 6 and 7 were created by extracting grains and noise from sample recordings of 3 seconds (water) or 5 seconds (rain).

For both sounds grains were normalized after extraction and attenuated by a random factor during synthesis. The random attenuation factors were drawn from a normal distribution with $\mu = 0$ and $\sigma = 3$, which also yielded negative attenuation factors. This increased the variation of the signal, since the amplitude of some grains was inverted.

To recreate a sound close to the original, the number of grains per second was set to 100, which corresponded approximately to the number of grains that was extracted from one second of the original sounds.

5.2. Morphing

The proposed algorithm works well for morphing between different sounds of rain or running water and yields realistic sounding results even when grains are extracted from only short sounds of only a few seconds length.

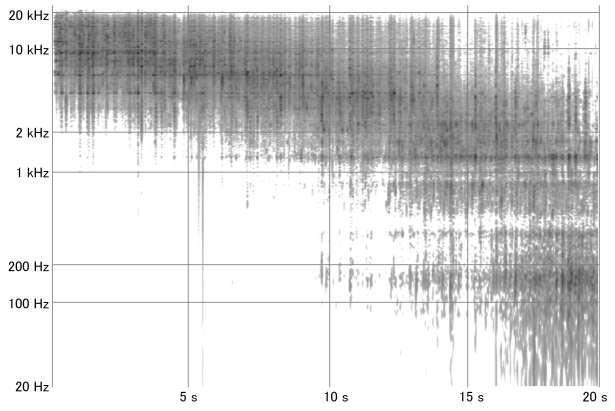

Figure 8: *Spectrogram of two sounds being morphed. Note how the formant slides between the positions in both sounds.*

Figure 8 shows the morph between two sounds synthesized from a recording of glass shards falling on a hard surface and a bubbling thick liquid. The spectral energy distribution is gradually changing between both sounds.

Morphing between very different sounds, like rain and fireworks, does not give very realistic results. However, this does not necessarily highlight a flaw in the proposed method, since such sounds do not exist in nature either.

5.3. Sound samples

Sound samples for synthesized sounds and morphed sounds can be found online[1].

6. FUTURE WORK

Apart from some necessary quality improvements in the grain extraction algorithm, there is much scope for enhancing the extraction of other features of the granular source sounds. These include the actual number of grains per second, the amplitude distribution of the grains or the temporal distribution of grains.

To consider granular sounds in which features change with time, like water splashes or breaking objects, temporal changes in the extracted parameters also need to be extracted.

7. REFERENCES

[1] Sadjad Siddiq, ``Morphing of impact sounds," in *Proceedings of the 139th Audio Engineering Society Convention*. Audio Engineering Society, 2015, to be published.

[2] William Moss, Hengchin Yeh, Jeong-Mo Hong, Ming C Lin, and Dinesh Manocha, ``Sounding liquids: Automatic sound synthesis from fluid simulation," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 3, pp. 21, 2010.

[1] See http://www.jp.square-enix.com/info/library/private/granularMorphing.zip. The password of the zip-file is "gmorph2015".

[3] Sadjad Siddiq, Taniyama Hikaru, and Hirose Yuki, ``当たって砕けろッ！プロシージャルオディ制作 (Go for broke! Creation of procedural audio content)," Presentation at the Computer Entertainment Developers Conference, 2014, Slides and sound samples are available at `http://connect.jp.square-enix.com/?p=2639` (visited on 25.9.2015).

[4] Curtis Roads, *Microsound*, MIT press, 2004.

[5] Øyvind Brandtsegg, Sigurd Saue, and Thom JOHANSEN, ``Particle synthesis--a unified model for granular synthesis," in *Proceedings of the 2011 Linux Audio Conference. (LAC'11)*, 2011.

[6] Charles Bascou and Laurent Pottier, ``New sound decomposition method applied to granular synthesis," *in ICMC Proceedings*, 2005.

[7] Jung-Suk Lee, François Thibault, Philippe Depalle, and Gary P Scavone, ``Granular analysis/synthesis for simple and robust transformations of complex sounds," in *Audio Engineering Society Conference: 49th International Conference: Audio for Games*. Audio Engineering Society, 2013.

[8] Diemo Schwarz, Roland Cahen, and Sam Britton, ``Principles and applications of interactive corpus-based concatenative synthesis," *Journées d'Informatique Musicale (JIM), GMEA, Albi, France*, 2008.

[9] Diemo Schwarz, Grégory Beller, Bruno Verbrugghe, Sam Britton, et al., ``Real-time corpus-based concatenative synthesis with catart," in *Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx), Montreal, Canada*. Citeseer, 2006, pp. 279--282.

[10] Lie Lu, Liu Wenyin, and Hong-Jiang Zhang, ``Audio textures: Theory and applications," *Speech and Audio Processing, IEEE Transactions on*, vol. 12, no. 2, pp. 156--167, 2004.

[11] Martin Fröjd and Andrew Horner, ``Sound texture synthesis using an overlap-add/granular synthesis approach," *J. Audio Eng. Soc*, vol. 57, no. 1/2, pp. 29--37, 2009.

[12] Saeed V Vaseghi, *Advanced digital signal processing and noise reduction*, John Wiley & Sons, 2008.

[13] Hermann Ludwig Ferdinand von Helmholtz, *Die Lehre von den Tonempfindungen als physiologische Grundlage für die Musik*, Vieweg, 1863.