GRANULAR ANALYSIS/SYNTHESIS OF PERCUSSIVE DRILLING SOUNDS

Rémi Mignot
IRCAM & CNRS, UMR 9912,
Analysis & Synthesis Team,
Paris, France
remi.mignot@ircam.fr

Ville Mäntyniemi, & Vesa Välimäki
Aalto University,
Dept. Signal Processing and Acoustics,
Espoo, Finland
v.mantyniemi@gmail.com
vesa.valimkai@aalto.fi

ABSTRACT

This paper deals with the automatic and robust analysis, and the realistic and low-cost synthesis of percussive drilling like sounds. The two contributions are: a non-supervised removal of quasi-stationary background noise based on the Non-negative Matrix Factorization, and a granular method for analysis/synthesis of this drilling sounds. These two points are appropriate to the acoustical properties of percussive drilling sounds, and can be extended to other sounds with similar characteristics. The context of this work is the training of operators of working machines using simulators. Additionally, an implementation is explained.

1. INTRODUCTION

This work is a part of a project[1] which aimed at improving the sound synthesis of working machines, in term of realism and computer resource consumption. The context is the safety at work, by the improvement of the operator training using realistic machine simulators. Among the numerous kinds of sounds, one focused our attention, the percussive drilling sounds of rig machines, and is the subject of this paper.

Two complementary algorithms are described here: a quasi-stationary noise removal using a method based on the Non-negative Matrix Factorization, and a granular analysis/synthesis method for an efficient extraction of individual clicks (shock sounds). This work is not especially dedicated to percussive drilling sounds, and it can be extended to all other sounds which have the same acoustical properties. But, because it took part in a given project, all the tests and results have been done with drilling sounds only.

The synthesis process used in this paper has similarities to other physically insprired percussive sound synthesis methods based on individual event generation [1, 2, 3, 4], but the way we create the event sounds is novel.

The drilling of the machines under interest, roughly consists in the repetitive percussions of a hammer against the rock to perforate. This hammer is at the extremity of a drill string composed by several connected rods, allowing to dig deep into the ground. The movement of this assembly of bars is fed by a hydraulic system. See Fig. 1 for an illustration of a drilling machine under interest.

Because different drilling situations may occur, different kinds of sounds can be heard. Here is a list of possible situations:

- *Normal drilling*: the typical drilling sound which should be heard when drilling.
- *Underfeed*: feeding is the method of pushing the drilling tool against the rock. In an underfeed situation, the drill is not pushed hard enough against the rock and the sound is altered compared to a normal drilling situation.
- *Overfeed*: overfeeding is caused by pushing the drill too hard against the rock, and this causes a slight variation in sound compared to normal drilling.
- *Closed Rattling*: rattling is the sound of the rods being removed from the rod string. The rods are shook heavily to open the threads holding the rods together. In this situation the threads are still closed causing a clearly distinguishable sound.
- *Open Rattling*: this situation occurs when the threads finally open is a very short but extremely loud and high frequency sound event, which is clearly different compared to rattling with the threads still closed.

Moreover, the heard sound may also depend on the length of the rod string, and the nature of the drilled rock.

Therefore, since the sounds vary with the situation, the operator may use this additional information to improve the perception of what is happening. Consequently, in the context of training and safety, the realism of the drilling sound synthesis of the rig simulator is significant. For example, the overfeeding which may cause a break of the rods, should be recognizable by the operator, nevertheless we know this situation is not easy to detect, cf. [5].

In this work, because of the numerous different possible sounds, and the numerous recorded sounds (almost one thousand), we have chosen to develop an automatic method, as robust as possible for all possible situations.

This paper is organized as follows: in sec. 2 the background noise removal is detailed. The different steps of the analysis/synthesis method are explained in sec. 3. Then, section 4 gives a brief overview of the software developed during the project. Finally, this paper is concluded in sec 5.

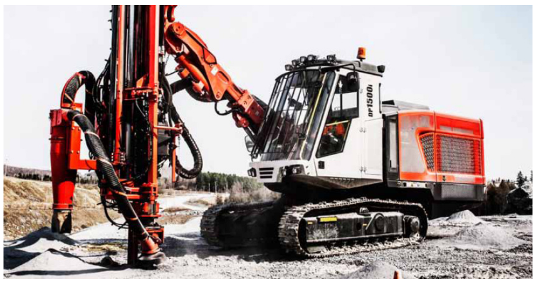


Figure 1: Drilling machine of Sandvik. Picture retrieved from the web site: http://construction.sandvik.com

[1]REMES project, funded by the Finnish Work Environment Fund TSR: project no. 113252.

2. BACKGROUND NOISE REMOVAL

All the available recordings of drilling sounds were corrupted by an inherent background noise coming from the diesel engine or the hydraulic system. Because of the different natures of this background noise and the drilling sound of interest, it is not possible to simultaneously analyze and synthesize them as a unique entity.

To analyze a noisy recorded sound and to resynthesize the noiseless sound, a first approach may be possible when the sound of interest can be relevantly modeled, and if the analysis is robust to noise, cf. e.g. [6]. But in our case, it seems really difficult to define a fine modeling of drilling sounds. Otherwise, an adaptive spectral subtraction may be possible to remove the background noise, cf. e.g. [7]. Nevertheless, it requires the knowledge of the noise spectrum which should be constant in time, stationary. Unfortunately, in our case it slowly changes because for instance of the moving engine speed, RPM (Revolutions Per Minutes).

Here, we propose an approach based on the Non-negative Matrix Factorization, which only assumes a quasi-stationary background noise, which may slowly change in time, and an approximatively known Signal-to-Noise Ratio (SNR).

2.1. Non-negative Matrix Factorization

For audio applications, the Non-negative Matrix Factorization (NMF) is usually used for polophonic music transcription, cf. e.g. [8, 9], denoising and source separation, cf. e.g. [10, 11]. It allows to approximate the matrix of the magnitude spectrogram, cf. e.g. [12, 8]. Using a Short-Time Fourier Transform (STFT), cf. e.g. [13, p.35], the $(M \times N)$ spectrogram matrix $V = |X|$ represents the N successive "instantaneous" magnitude spectra of the sound, which are given by its columns with M frequencies. Using this time-frequency representation, we know the variation in time of the frequency components.

In this case, the Non-negative Matrix Factorization consists in the approximation of the $(M \times N)$ matrix V, which has only non-negative elements, into the product WH as follows:

$$V \approx WH \quad \Longleftrightarrow \quad V_{m,n} \approx \sum_{k=1}^{K} W_{m,k} H_{k,n}, \tag{1}$$

where W is a $(M \times K)$ matrix and H is a $(K \times N)$ matrix. The dimension K of the factorization is chosen much smaller than M and N, i.e. $K \ll M$ and N.

Consequently, the Non-negative Matrix Factorization models the columns of V as a weighted sum of the columns of W. As a first conclusion, W is considered as the frequency dictionary giving the basis with size K on which the spectrogram V is decomposed. And since the kth row of the matrix H gives the time-varying weight of the kth word of W, column k, the matrix H is considered as the time-activation matrix.

Basically, the algorithm of this factorization consists in the minimization of a "distance" between the original spectrogram V and its approximation $\widetilde{V} = WH$. Two standard choices are the well-known Euclidean distance, and the generalized Kullbach-Leibler divergence more common in NMF, cf. e.g. [14].

The solving of this minimization problem is usually based on the iterative Newton algorithm, cf. e.g. [15, ch.4.3]. Starting from initial non-negative matrices W and H, usually randomly chosen, the matrices W and H are successively and iteratively updated. This algorithm proves to converge to the closest local minimum of the initial values. Note that we have used for this work the generalized Kullbach-Leibler divergence.

Note that additionally, the columns of W are normalized, without affecting the modeling. With $\lambda_k = \sum_m W^2_{m,k}$, the norms of the columns of W, and $\Lambda = \mathrm{diag}([\lambda_1, \lambda_2, \dots \lambda_K])$, we apply the assignments: $W \leftarrow W\Lambda$ and $H \leftarrow \Lambda^{-1}H$, at every iteration.

2.2. Quasi-stationary noise removal using NMF

As said previously, the value $H_{k,n}$ of the activation matrix gives the contribution of the kth column of W at the time index n. Then, if the kth row of H slowly varies in time, the contribution of the kth word of W also varies slowly. To remove the background noise assuming that it is quasi-stationary, the basic idea of our approach is to constrain some rows of H to vary slowly. Note that, this time smoothing is a common used constraint with NMF, cf. e.g. [10, 9, 16]. The proposed method of this paper is noticeably simpler, but proves to be satisfying in our case.

Let's define K_n the size of the noise basis, the modified NMF algorithm for the background noise removal consists in adding a "smoothing" operation of the K_n first rows of H after its updates. As a result, this new iterative algorithm will implicitly learn the noise basis, which is stored in the corresponding K_n first columns of W. At the same time, since the properties of the drilling sound are opposite, this process also learns the sound basis, which is stored in the remaining $K - K_\mathrm{n}$ other columns of W.

The smoothing operation of the first rows of H only consists in a linear filtering with a very low cutoff frequency f_c. For example, if $f_c = 2$ Hz, the obtained noise is authorized to vary only twice a second. In the same time, if $K_\mathrm{n} = 2$, in principle this method tries to extract a slowly time-varying noise, by modeling it as a moving mix of two different noises.

Nevertheless, as such, this approach does not succeed to remove the background noise, because at convergence, the gain of the corresponding rows of H are too low, and the whole spectrogram V is actually modeled by the other $K - K_\mathrm{n}$ words. To solve this problem we add a second constraint: choosing the value σ of the Signal-to-Noise Ratio (SNR), which is the ratio between the energy of the drilling sound and the energy of the background noise, we artificially modify the gains such that this ratio is checked, and without modifying the total energy.

Let's define $E\{X\} = \sum_{m,n} X^2_{m,n}$ the energy operator for all $(M \times N)$ matrices, V_n and V_s the spectrogram of the noise and the signal respectively, and the decomposition $H^T = [H_\mathrm{n}^T, H_\mathrm{s}^T]$ with $.^T$ the matrix transpose. The additional constraints are then:

$$E\{V_\mathrm{s}\} + E\{V_\mathrm{n}\} = E\{V\}, \tag{2}$$

$$E\{V_\mathrm{s}\} \;/\; E\{V_\mathrm{n}\} = \sigma. \tag{3}$$

With $\alpha_\mathrm{n} = E\{V\}/(E\{V_\mathrm{n}\}(\sigma+1))$ and $\alpha_\mathrm{s} = \alpha_\mathrm{n}\sigma E\{V_\mathrm{n}\}/E\{V_\mathrm{s}\}$, these constraints are verified by assigning after every normalization $H_\mathrm{n} \leftarrow \alpha_\mathrm{n} H_\mathrm{n}$ and $H_\mathrm{s} \leftarrow \alpha_\mathrm{s} H_\mathrm{s}$.

Because it may be difficult to know the "real" SNR of the input noisy sound, it seems to be interesting to release the SNR adjustment. For that, first the iterative algorithm is computed with Q_1 iterations where both constraints are applied, smoothing and gain adjustment, then Q_2 additional iterations are computed with only the smoothing. Consequently, during the Q_1 first iterations, the noise dictionary is learned by forcing the SNR, then the algorithm continues from this solution and converges to a close solution, with a possible different SNR.

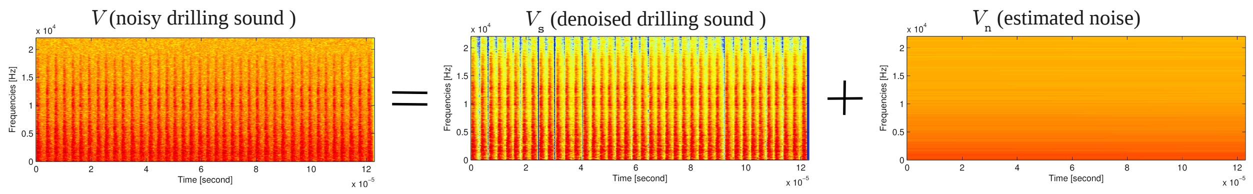


Figure 2: Illustration of the NMF based denoising. Spectrograms of the input noisy sound, denoised drilling sound, and reconstructed noise.

2.3. Noiseless drilling sound reconstruction

With the obtained approximation $\widetilde{V} = WH$, and with the original phase matrix $\Phi = \angle X$, the time signal is reconstructed with the inverse Short-Term Fourier Transform of $\widetilde{X} = \widetilde{V} \otimes \mathrm{e}^{j\Phi}$, with $\otimes$ the element-wize product. Then, the derivation of the estimated noiseless drilling sound y, follows the same principle: the inverse Short-Term Fourier Transform of $Y = V_n \otimes \mathrm{e}^{j\Phi} = (W_\mathrm{n} H_\mathrm{n}) \otimes \mathrm{e}^{j\Phi}$ is computed. This corresponds to a non-supervised source separation where the drilling sound of interest and the background noise are considered as two distinct sources.

For the project we used the following parameters: first the sampling rate F_s is forced by the available recordings, around 48 kHz. The Short-Term Fourier Transform is computed using a Hann sliding window with size 1024 samples and a fine hop size of 128 samples. The bilateral spectra with size 2048 are reduced to only consider the frequency range $[0, F_s/2]$, then $M = 1025$ bins. The NMF dimension is $K = 82$, and the noise dimension is $K_\mathrm{n} = 2$, using the cutoff frequency $f_c = 2$ Hz. During the $Q_1 = 100$ first iterations, the SNR has been constrained to 1 (0dB), and the algorithm continues with $Q_2 = 100$ other iterations. See for example the illustration of Fig. 2.

3. DRILLING SOUND ANALYSIS/SYNTHESIS

Having the denoised drilling sounds, the new challenging task is to analyze them in order to synthesize them using a realistic and very low-cost method, but also with the possibility to change the mean frequency and other parameters.

Remark that the sound is physically produced by the shock of the hammer against the rock, with possible resonances along the rod. Because of the irregular repetition of the shocks, in term of time position, amplitude, and spectrum, we choose a granular approach, which seems well suited for this case, cf. e.g. [17]. In the following, single strike sounds from a rock drill will henceforth be referred to as clicks.

To summarize the analysis process, the shocks instant and the amplitude envelope of the clicks in time are alternatively estimated twice. The first stage uses a robust but inaccurate technique, and the second one is based on a refined optimization procedure which requires a relatively good initialization. Then, some clicks are individually extracted from the input denoised sound, and the synthesis only consists in repetitively playing these extracted clicks. To get a natural synthesis, the irregularity is reproduced by randomly choosing the clicks among the stored click collection, for the spectral irregularity, and by randomly choosing the time positions and the amplitudes. These two last random processes are done according to simple rules which are learned with the input drilling sound, mean and variance of the frequency and the amplitude.

In this section, we first give some tools for the time-envelope estimation of the drilling sounds, and then all stages of the analysis are successively described.

3.1. Time-envelope estimation

Integrated energy: To detect the click positions in time and their amplitudes, we define here a smooth time function based on energy integration. With x_n the denoised drilling sound at sample n, w_n a finite weighting window with size $2N+1$, we define the smooth energy function e_n as follow:

$$e_n = \sum_{j=-N}^{N} x_{n+j}^2 w_j^2 \tag{4}$$

This energy function has the property to efficiently smooth the signal and to raise the clicks as obvious maxima. Then it will be easier to detect the click occurrences by analyzing the peaks of e_n than analyzing the peaks of x_n. Remark that as seen below, the window shape w_n is an important feature in the click detection, and it will be refined later.

Whitening: At first look, as such the energy function e_n has obvious peaks which directly correspond to the click occurrences, but has also many local maxima which do not correspond to a click.

For this reason, the original denoised drilling sound is first *whitened* by filtering it by the inverse filter obtained by a linear prediction, cf. [18]. This operation provides a deconvoluted signal y_n which has a flat spectral envelope.

Instead of computing the energy function e_n with the signal x_n, we analyze its white version y_n. This inverse filtering has the property to remove the minimal phase part, and in some cases, it concentrates the energy of a single click at its beginning. Moreover, when the high frequency components are low, it raises them, then the *contours* are enhanced for a better precision.

Here the chosen LPC order is 1024, which is quite high and provides an expensive filter, but this process is computed off-line during the analysis only.

Modeling the time-envelope of clicks: We assume here that all single clicks are the product of an unknown flat signal, possibly stochastic and different for every click, and of an envelope, identical for all clicks. This envelope is now modeled using an Attack/Decay model, separated into two parts:

- The increasing attack with length N_a and parameter α_a. Its formula $\forall n \in [0, N_a]$ is

$$a_n = \begin{cases} n/N_a, & \text{if } \alpha_a = 0, \\ \frac{1-\exp(-\alpha_a n)}{1-\exp(-\alpha_a N_a)}, & \text{otherwise.} \end{cases} \tag{5}$$

  Note that this envelope increases from 0 at $n = 0$ to 1 at $n = N_a$. An example of this behavior is shown in Fig. 3.

- The decreasing decay part has an exponential behavior with α_d the positive damping factor. Its formula $\forall n > N_a$ is

$$a_n = \exp\left(-(n - N_a)\alpha_d\right). \qquad (6)$$

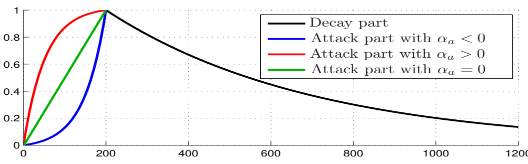


Figure 3: Illustration of an individual click envelope with different values of α_a. Here the attack time is $N_a = 200$ samples.

In the following sections, the full procedure for the click detection and the envelope estimation are described.

3.2. First estimation

First click detection: Based on the estimated energy function computed with the white signal y_n, and with a Hann weighting window with length 1023 samples, a first click detection is done by picking up the local maxima of e_n. Also, in order to know the limits of the clicks, the estimated click position are associated to the pair of local minima on the left and on the right. Then, for all detected clicks, we have: the positions P_p of the peak, P_l and P_r of the closest minimum on the left and on the right respectively.

First estimation of the envelope parameters: The previous window size is chosen to eliminate enough false alarms, but the obtained energy function is too smooth. Then, a new energy function e'_n is computed with a smaller window with size 511 samples and the envelope parameters α_a, N_a and α_d are estimated by matching the modeled envelope a_n to the square root of e'_n: $\nu(n) = \sqrt{e'_n}$, on all time ranges $[P_l, P_r]$, for all detected clicks, cf. Fig. 4.

- N_a: the attack time is computed as the median value of all $P_p - P_l$, for all detected clicks.
- α_d: the estimation of the decay factor is straightforward. Using all the values $\nu(P_p)$ and $\nu(P_r)$, for all clicks, all α_a are obtained solving

$$\begin{aligned} \nu(P_r) &= \mathrm{e}^{-\alpha_d(P_r - P_p)}\,\nu(P_p) \\ \Leftrightarrow \qquad \alpha_d &= \frac{-1}{(P_r - P_p)} \log \frac{\nu(P_r)}{\nu(P_p)}, \end{aligned}$$

  and the used value is the mean of all computed values.
- α_a: the estimation of the attack factor is not easy because it is not possible to isolate it from eq. (5); we use an iterative procedure to solve the problem. With $P_m = (P_p + P_l)/2$ the middle point between P_p and P_l, the obtained attack must join the minimum point in P_l and the maximum point in P_p passing through the middle point in P_m, which means that the following equation must be solved:

$$\nu(P_m) = a(P_m - P_l)\,\nu(P_p).$$

  This solving iterative procedure is based on the simplex method, cf. [19].

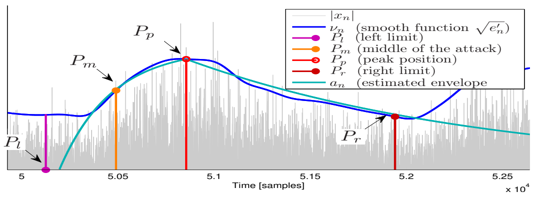


Figure 4: Illustration of the first envelope estimation. Here N_a, α_a and α_d are estimated such that the envelope of all individual clicks passes close to the 4 drawn points.

3.3. Second click detection

In the previous section, the click detection and the envelope modeling were based on the calculus of the energy functions e_n and e'_n. Unfortunately, this click detection is highly biased, because of the used window w_n which is a Hann window.

But, having a coherent estimation of the click envelope a_n we can significantly reduce this bias by replacing the Hann window by the click envelope itself. Then a new refined energy function e_n is computed using $w_n = a_n$, and again, the click positions in time are detected using the local maxima of e_n.

Indeed, if the (denoised) white signal y_n is the product of a stationary signal u_n with variance σ^2, and an envelope signal g_n which consists in a succession of click envelopes a_n, i.e. $g_n = \sum_i \beta_i a_{n-P_i}$ with P_i the time position of the ith click and β_i its amplitude, the energy function is written

$$e_n = \sum_{j=-N}^{N} y_{n+j}^2 a_n^2 = \sum_{j=-N}^{N} u_{n+j}^2 \Big(\sum_i \beta_i a_j a_{n+j-P_i}\Big)^2.$$

Since $a_n \geq 0$, the function $\left(\sum_i \beta_i a_j a_{n+j-P_i}\right)^2$ has local maxima at $n = P_i$, the true click positions, and also e_n because here u_n is assumed to be stationary.

Nevertheless, as such, this operation does not provide a good solution because of some reasons, and we need to add some limitations. First, as explained below, the first parameter estimation of the previous section is also biased, and usually the decay factor is over-estimated, then w_n is defined using the envelope a_n with a decay factor α_d half than the estimated value. Second, the size of the window $w_n = a_n$ is chosen to be half of the mean distance of the detected clicks in the previous section. This limitation reduces the influence of neighbor clicks, and improves the resolution. Third, the new energy function e_n has some "false" local minima. Then the search of the "true" local maxima is done by searching the absolute maxima of the new e_n over the time ranges: $[P_l - N_a, P_p - N_a]$, where P_l, P_r and N_a are the minima positions and the attack time estimated previously. Here the ranges are delayed by N_a, because now the maxima does not give the center of the click, but its beginning.

If no maximum is found in a selected range, then we assume there is no corresponding click, and the range is deleted. Among the obtained maxima, which are the estimated click positions P_c, beginning, we also delete some of them such that the minimal distance between two neighboring detected clicks is smaller than half of the median. The choice of the detections to delete depends on the positions themselves and also on the amplitudes e_{P_c}.

3.4. Second estimation of the envelope parameters

The estimation of section 3.2, based on a fitting of some points of the energy function e'_n, was also biased. Then, having the refined click positions P_c, with less false detections, now we can also refine the estimation of the envelope parameters N_a, α_a, and α_d.

We here propose a more accurate approach, based on an iterative algorithm for the solving of a non-linear optimization problem and with few parameters, three in our case. The chosen algorithm is based on the simplex method, cf. [19] and is implemented in Matlab as the standard function *fminseach()*.

To define the criterion to minimize, let's remark that: with a good estimation of the envelope a_n, dividing the signal y_n by the combined envelope $g_n = \sum_i \beta_i a_{n-P_i}$, with P_i the new refined position and $\beta_i = \sqrt{e(P_i)}$ its amplitude, the obtained signal z_n should have an envelope as flat as possible. Then the criterion C is based on the flatness of the energy function f_n computed with z_n. It is computed as follow:

- The envelope a_n of an individual click is computed using eqs. (5) and (6), for all $a_n \geq 0$.
- The "combined" envelope is given by: $g_n = \sum_i \beta_i a_{n-P_i}$.
- The signal z_n is computed as follows: $z_n = y_n/g_n$.
- The energy function f_n is computed with z_n and a Hann window w_n with size 1024: $f_n = \sum_{j=-N}^{N} z_{n+j}^2 w_j^2$.
- The flatness is now tested by computing the square sum of the difference of f_n and its mean. If z_n is flat, f_n is close to a constant function and the following criterion is small:

$$C = \sum_{n=0}^{N} \left(f_n - \frac{1}{N} \sum_{i=0}^{N-1} f_i \right)^2$$

With this definition, the simplex algorithm [19] provides the parameter values (N_a, α_a, and α_d) of the closest local minimal of C, from the initial values.

3.5. Last click detection

Again, with the refined envelope parameters we can obtain a refined detection of click positions, and their corresponding amplitudes as in sec. 3.3. But now, because the estimation of α_d is not biased, its value is not divided by 2.

Remark that, since the new click position P_c are refined, we could imagine to make an iterative process to refine again the envelope parameters, and so on. But in the most favorable cases, the improvements are insignificant, and in the worst cases, the process may be unstable, and may provide worse results. Then, the estimation of the click position P_c, their corresponding amplitude β_c, and the envelope parameters N_a, α_a and α_d stops here.

3.6. Click extraction

With all detected values P_c and β_c for all clicks, and the envelope parameters N_a, α_a and α_d, this section explains how a collection of some click is extracted from the denoised signal x_n. The total number of detected clicks is denoted C_d, and the number of extracted clicks is denoted C_e. Basically, $C_d \approx 70$ for 2 seconds of signals with a frequency of 35 clicks per second, and the chosen number of extracted clicks is $C_e = 10$.

The chosen extracted clicks must check the following constraints: first, to reduce the overlap of the next neighbor clicks, the distance between P_c and P_{c+1} must be as high as possible. Second, to reduce the contribution of neighbor clicks, on the left and on the right, its amplitude β_c must be higher than β_{c-1} and β_{c+1}. Then, all the C_d clicks are sorted according to these criteria, and the C_e preferred clicks are selected for extraction.

In a first step, the signal y_n is flattened, as in sec. 3.4. The flat signal z_n is obtained by the division of y_n by the new "combined" envelope

$$g'_n = \max_{i \in [1, C_d]} \Big(\beta_i a_{n-P_i} \Big).$$

Note that here we use the "max" operator instead of the sum operator. The use of the max avoids the "cumulation" of the tails of the envelopes, whereas the use of the sum favors this cumulation. In section 3.4, this effect is used to limit the value of α_d during the optimization, and now it is preferable to limit this effect.

As a result, the signal z_n is flat, which means that the effect of the click envelope is canceled, as shown in the middle sub-figure of Fig. 5. Now to extract an individual click with a damping tail which overlaps with the following clicks, we just have to multiply the flat signal z_n by the envelope a_n of an individual click, cf. the bottom sub-figure of Fig. 5.

In this way, neither the tail of the previous click is totally removed, and nor the signal of the following one, in a strict sense. But the contribution of neighbor clicks is efficiently removed for the following reasons: first, thanks to the simultaneous masking, the extracted click of interest efficiently masks the tail of the previous one because its envelope is smaller. Second, the amplitude of the following one is efficiently reduced and thanks to the temporal masking, its contribution is not audible in most of the cases.

Finally, since the clicks are associated with the white signal y_n, then we cancel this whitening by filtering the extracted click samples by the LPC filter $H(z) = 1/A(z)$, cf. sec. 3.1. Moreover, since the extracted clicks have different amplitudes, they are normalized to a unique norm.

3.7. Statistical parameters

The real-time granular synthesis only consists in successively playing the C_e extracted clicks with a random selection at each time and with a quasi-constant frequency. However, to provide a realistic synthesis, it is needed to add a fluctuation in frequency and in amplitude, as it is the case in real drilling sounds.

First the mean period T_0, in samples, of the click occurrences is computed as the mean of the distances of detected clicks:

$$T_0 = \frac{1}{C_d - 1} \sum_{i=1}^{C_d - 1} P_{i+1} - P_i,$$

and the mean frequency is then $F_0 = F_s/T_0$. Also, to take account of the variance of the click position, the fluctuation in term of frequencies, the standard deviation is computed as follows:

$$\sigma_T = \Big(\frac{1}{C_d - 1} \sum_{i=1}^{C_d - 1} \big(P_{i+1} - P_i - T_0 \big)^2 \Big)^{\frac{1}{2}}$$

Second, in the same way the mean amplitude B and its standard deviation are computed:

$$B = \frac{1}{C_d - 1} \sum_{i=1}^{C_d - 1} \beta_i$$

$$\sigma_B = \Big(\frac{1}{C_d - 1} \sum_{i=1}^{C_d - 1} \big(\beta_i - B \big)^2 \Big)^{\frac{1}{2}}$$

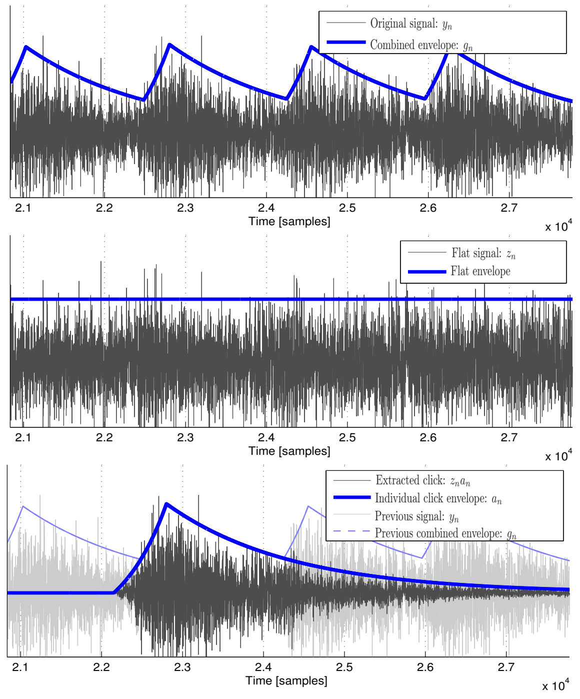


Figure 5: Illustration of the click extraction, based on the flattening of the termporal envelope of a drilling sound recording.

3.8. Real-time synthesis

Then, when a new individual click has just started to be played back, in real-time, the synthesizer chooses the instant of the next played click with a delay time randomly chosen according to a distribution with mean $T_0 = 1/F_0$ and variance σ_T^2. Also its amplitude is randomly chosen according to a distribution with mean B and variance σ_B^2. This procedure provides some fluctuations in the amplitudes and the positions which makes the synthesis more realistic. Moreover, the new played click is randomly chosen among the collection of the C_e extracted clicks. Figure 6 summarizes the global sound analysis/synthesis process.

In this work, the background noise of the engine is not resynthesized from the NMF analysis. The reasons are that its resynthesis based on the NMF does not yield a satisfying sound quality and that another good technique for background noise synthesis was already available in the project.

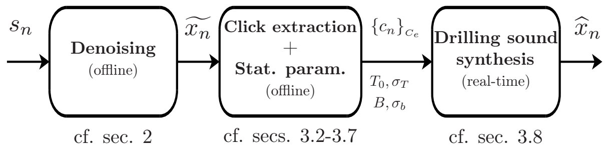


Figure 6: Summary of the global sound synthesis process: first the drilling sound of interest x_n is extracted from the noisy recorded sound s_n, cf. sec. 2, then the analysis extracts C_e clicks noted here c_n, cf. 3.2-3.6, and the statistical parameters (T_0, σ_T, B and σ_B) are estimated, cf. sec. 3.7. Finally the sound $\widehat{x_n}$ is synthesized, cf. 3.8.

3.9. Informal listening test

No formal and rigorous listening test has been done to evaluate the denoising and the synthesis. Nevertheless, the simple comparison of a former drilling sound synthesis, made by Creanex, and this new synthesis reveals an outstanding improvement. Moreover, an informal listening test has been made with four experts who have an excellent experience in drilling machines. As a result, they judged the denoised sounds as sufficiently correct, and they found the drilling synthesis realistic and accurate. Consequently, it proves the ability of this method to accurately synthesize drilling sounds. More details of this test are given in [20].

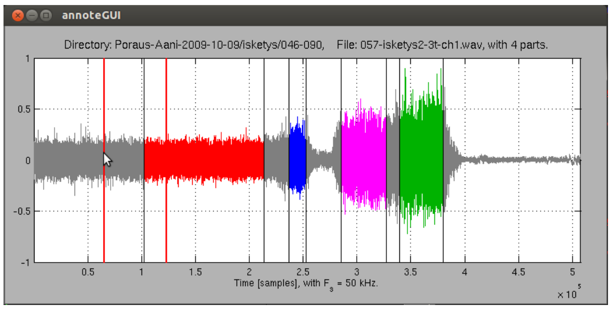
4. SOFTWARE IMPLEMENTATION

This section summarizes the software developed during the REMES project. The main goal is to provide some tools and examples for: the manual annotation of drilling sounds, the automatic analysis, the test of results, and the efficient real-time synthesis.

These applications have been used with the drilling sounds provided by Sandvik, but they have been designed to work with other sounds having similar signal properties. For more details about this software package, we refer the reader to [20].

4.1. Annotation

Because most of the recordings contain some different successive drilling situations (normal drilling, overfeed, rattling, ...), it is necessary to annotate the limits of each part for each file. The automatic annotation is difficult, and it was not the subject of this work. Moreover, a bad annotation may provide a worse analysis. Then a simple Graphical User Interface has been developed in Matlab to annotate easily and quickly all the sounds, cf. Fig. 7.


annoteGUI

Directory: Poraus-Aani-2009-10-09/isketys/046-090, File: 057-isketys2-3t-ch1.wav, with 4 parts.

Figure 7: Screenshot of the annotator software. In this example, 4 parts are annotated with different colors.

4.2. Analyzer

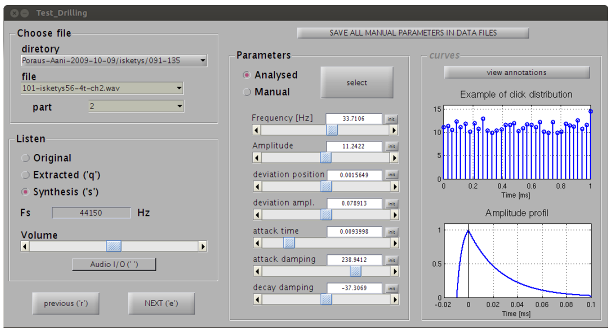
The analyzer is a piece of software which automatically computes the analysis of all annotated parts, without manual intervention.

At the moment of the end of the project, 1296 parts are annotated, and the complete analysis lasts approximately 7 hours, using a 4 core CPU at 3.20 GHz. All steps of sec. 3 are computed successively for all annotated parts. Note that to make this process faster, the length of the parts are limited by 2 seconds. Finally, all analyzed parameters, such as the extracted clicks, the envelope parameters N_a, α_a and α_d, and the statistical parameters, F_0, σ_T, B and σ_B, are stored into some data files.

4.3. Test and parameter refinement

To check and eventually modify the analyzed parameters, another GUI has been developed using Matlab, cf. Fig. 8.

This software loads the analyzed drilling data, computed by the analyzer, and proposes to compare the original noisy sound, the denoised drilling sound and the synthesis. Also, some sliders and editor controls allow to modify the synthesis parameters, and eventually to save the results in the data file.



Choose file
diretory: Poraus-Aani-2009-10-09/isketys/091-135
file: 101-isketys56-4t-ch2.wav
part: 2

Listen
Original
Extracted ('q')
[✓ Synthesis ('s')]
Fs: 44150 Hz
Volume

Parameters
[✓ Analysed]
Manual
Frequency [Hz]: 33.7106
Amplitude: 11.2422
deviation position: 0.0015649
deviation ampl.: 0.078913
attack time: 0.0093998
attack damping: 238.9412
decay damping: -37.3069

Figure 8: Screenshot of the Graphical User Interface for testing and for the parameter modification.

4.4. Real-time synthesis library

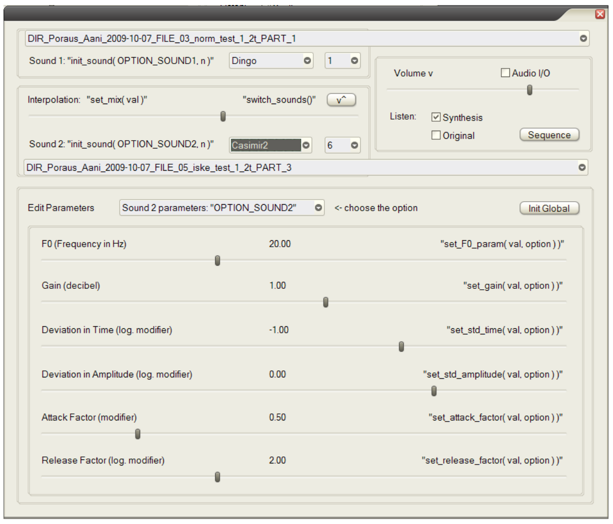
For the efficient real-time synthesis, a C++ library and a demonstration application have been developed. The main benefit of this drilling sound synthesis tool, compared to a simple playback of the denoised sound, is the ability to modify in real-time some parameters and to reproduce a continuous modification of the sound.

The available parameters are: the mean frequency, the mean amplitude, the deviation in time, the deviation in amplitude, a modified attack factor and a modified decay factor. Fig. 9 shows a screenshot of this demonstration application, with six sliders for the parameter control.

5. CONCLUSION

These paper presents two complementary methods useful for the purpose of the mentioned project: first, a non-supervised removal of quasi-stationary noise has been proposed, which is based on the Non-negative Matrix Factorization. This algorithm is relevant because of the opposite natures of the background noise and the drilling sound of interest. Second, a granular analysis/synthesis approach has been presented. It roughly consists in: the estimation of the click positions in time and of the time envelope modeling of the single clicks; the extraction of some individual clicks; and finally a successive play of the stored clicks, with some random rules learned on the original sound. Not only the synthesis method used a small amount of memory, less than 500 kBytes per sound, but also the CPU consumption is negligible on current personal computers.

Note that, at the end of this work, 1296 annotated drilling sounds have been analyzed for tests. These recorded sounds have



DIR_Poraus_Aani_2009-10-07_FILE_03_norm_test_1_2t_PART_1
Sound 1: "init_sound(OPTION_SOUND1, n)": Dingo, 1
Volume v
Audio I/O
Interpolation: "set_mix(val)"
"switch_sounds()"
Listen: [✓ Synthesis] Original
Sound 2: "init_sound(OPTION_SOUND2, n)": Casimir2, 6
DIR_Poraus_Aani_2009-10-07_FILE_05_iske_test_1_2t_PART_3

Edit Parameters: Sound 2 parameters: "OPTION_SOUND2"

F0 (Frequency in Hz)	20.00	"set_F0_param(val, option))"
Gain (decibel)	1.00	"set_gain(val, option))"
Deviation in Time (log. modifier)	-1.00	"set_std_time(val, option))"
Deviation in Amplitude (log. modifier)	0.00	"set_std_amplitude(val, option))"
Attack Factor (modifier)	0.50	"set_attack_factor(val, option))"
Release Factor (log. modifier)	2.00	"set_release_factor(val, option))"

Figure 9: Screenshot of the Graphical User Interface. Example of implementation of the C++ real-time library.

been produced by rig machines of Sandvik during previous projects, cf. e.g. [5]. The developed methods and software package can be used for synthesizing realistic drilling sounds in a working machine simulator. As demonstration, the companion webpage [21] proposes the listening of some excerpts of sounds: original, denoised and synthesized.

6. ACKNOWLEDGMENTS

This work was funded by the Finnish Work Environment Fund TSR (project no. 113252, REMES – Realistic Machine and Environmental Sounds for a Training Simulator to Improve Safety at Work).

7. REFERENCES

[1] P.R. Cook, "Modeling Bill's gait: Analysis and parametric synthesis of walking sounds," in *Proc. Audio Eng. Soc. 22nd Conf. Virtual, Synthetic, and Entertainment Audio*, 2002, pp. 73–78.

[2] L. Peltola, C. Erkut, P.R. Cook, and V. Välimäki, "Synthesis of hand clapping sounds," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 15, no. 3, pp. 1021–1029, 2007.

[3] R. Nordahl, L. Turchet, and S. Serafin, "Sound synthesis and evaluation of interactive footsteps and environmental sounds rendering for virtual reality applications," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 9, pp. 1234–1244, 2011.

[4] S. Oksanen, J. Parker, and V. Välimäki, "Physically informed synthesis of jackhammer tool impact sounds," in *Proc. Int. Conf. Digital Audio Effects (DAFx-13)*, Maynooth, Ireland, Sept. 2013, pp. 168–171.

[5] S. Oksanen, T. Pirinen, and V. Välimäki, "Subjective assessment of percussive drilling sounds," in *Proc. Internoise 2009*, Ottawa, Canada, Aug. 2009, 2009.

[6] C. Févotte, B. Torrésani, L. Daudet, and S.J. Godsill, "Sparse linear regression with structured priors and application to denoising of musical audio," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 1, pp. 174–285, 2008.

[7] S.F. Boll, "Suppression of acoustic noise in speech using spectral subtraction," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-27, no. 2, pp. 113–120, Apr. 1979.

[8] P. Smaragdis and J.C. Brown, "Non-negative matrix factorization for polyphonic music transcription," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, USA, Oct. 2003.

[9] N. Bertin, R. Badeau, and E. Vincent, "Enforcing harmonicity and smoothness in bayesian non-negative matrix factorization applied to polyphonic music transcription," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 638–549, 2010.

[10] T. Virtanen, "Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria," *IEEE Trans, Audio, Speech, and Language Processing*, vol. 15, no. 3, pp. 1066–1074, Mar. 2007.

[11] A. Ozerov and C. Févotte, "Multichannel nonnegative matrix factorization in convolutive mixtures for audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 550–563, 2010.

[12] D.D. Lee and H.S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[13] U. Zölzer (Ed.), *DAFX - Digital Audio Effects*, John Wiley & Sons, Ltd., 2002.

[14] D.D. Lee and H.S. Seung, "Algorithms for non-negative matrix factorization," *Advances in Neural Information Process. Systems*, pp. 556–562, 2000, MIT Press.

[15] E. Walter and L. Pronzato, *Identification of Parametric Models*, Communications and Control Engineering, 1997, 413 pages.

[16] N. Mohammadiha, P. Smaragdis, and A. Leijon, "Supervised and unsupervised speech enhancement using nonnegative matrix factorization," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 10, pp. 2140–2151, 2013.

[17] B. Truax, "Real-time granular synthesis with a digital signal processor," *Computer Music Journal*, vol. 12, no. 2, pp. 14–26, 1988.

[18] J. Makhoul, "Linear prediction: A tutorial review," *Proceedings of the IEEE*, vol. 63, 1975.

[19] J.C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence properties of the Nelder-Mead simplex method in low dimensions," *SIAM Journal of Optimization*, vol. 9, no. 1, pp. 112–147, 1998.

[20] V. Mäntyniemi, R. Mignot, and V. Välimäki, "REMES Final Report - The Finnish Work Environment Fund TSR Project no. 113252," Series: Aalto university publication series science + technology, 16/2014, Aallto University, 2014.

[21] R. Mignot, "Granular analysis/synthesis of precussive drilling sounds: Companion web page," `http://research.spa.aalto.fi/publications/papers/dafx15-drilling/`.

[22] M. Heiniö, "Rock excavation handbook," Sandvik Tamrock Corp, 1999.

[23] V. Mäntyniemi, "Sound synthesis in working machine simulators," M.S. thesis, Aalto University, 2014.