Mathematical Morphology Applied to Feature Extraction in Music Spectrograms

Gonzalo Romero-García, Isabelle Bloch, Carlos Agón

Mathematical Morphology Applied to Feature Extraction in Music Spectrograms*

Gonzalo Romero-García[1][0000−0003−1971−6204], Isabelle Bloch[2][0000−0002−6984−1532], and Carlos Agón[1][0000−0001−7926−2537]

[1] Sorbonne Université, CNRS, IRCAM, STMS, Paris, France
{romero,agonc}@ircam.fr
[2] Sorbonne Université, CNRS, LIP6, Paris, France
isabelle.bloch@sorbonne-universite.fr

Abstract. Mathematical Morphology has proven to be a powerful tool for extracting geometric information from greyscale images. In this paper, we demonstrate its application to spectrograms (two-dimensional greyscale images of sound) of music excerpts. The sounds of musical instruments exhibit particular shapes when represented as a spectrogram. These shapes are determined by the sound characteristics. In general, musical sounds contain three different components: the attack component, appearing as vertical lines; the sustain component, appearing as horizontal lines; and the stochastic component, appearing as a landscape of hills and holes. In this paper we propose a pipeline of morphological operators to separate these three components. This separation allows us to build a new sound similar to the input one.

Keywords: Mathematical Morphology · Spectrograms · Feature Extraction · Image Analysis.

1 Introduction

Mathematical Morphology (MM) has been proven to be a useful tool for image analysis and segmentation. In this work, we show how to apply it to a particular type of images: music spectrograms. Spectrograms are the most popular time-frequency representations of audio [9, p. 23]. Since they present the audio content as a function with time and frequency as domain and intensity as codomain, they can be viewed as greyscale images.

Audio signals exhibit various shapes when represented through a spectrogram. However, in general, music instruments have a particular way of being laid down inside a spectrogram; we can, in most of the cases, find a combination of three main components: the sinusoidal component, the transient component and the noise component. These three components appear with characterizing

* This work was partly supported by the chair of I. Bloch in Artificial Intelligence (Sorbonne Université and SCAI).

shapes: the sinusoidal component appears as horizontal lines, the transient component appears as vertical lines, and the noise component appears as a landscape of hills and holes (see Figure 1).

In order to detect these components and to be able to characterize the sound, we need to extract some features that allow us to recompose the signal; in the case of the lines (*i.e.*, the sinusoidal and transient components) we want to obtain the times and frequencies (that is, the pixels) that characterize the lines, alongside with the amplitude (*i.e.*, the greyscale level). In the case of the noise, since it is a stochastic component, we do not want the precise values of the hills and the holes. We rather want a mask to be applied to a white noise through a pointwise multiplication. With these data, we are able to reconstruct the sound with the help of a synthesis model.

In this work, we propose a pipeline of MM operators to extract these components. MM operators are well suited to detect the lines in the spectrogram and also to remove them such that a mask for the white noise is obtained.

The paper is organized as follows: we first expose the related work in spectrogram feature extraction in Section 2; then, we expose the mathematical definition of spectrograms in Section 3. After this is set up, we propose our morphological pipeline in Section 4 and show the results of our experiments in Section 5. We expose our conclusions and future work in Section 6.

2 Related work

The feature extraction from a spectrogram of audio was impulsed by the work of X. Serra [15,16], where a model of sound synthesis was proposed, based on a sum of a sinusoidal component plus a stochastic component, and peak detection methods were used to perform the analysis. Later, T. Verma and T. Meng proposed a sinusoids plus transients plus noise (STN) model in [18], that is the model we use in this work.

There has been a lot of work in the estimation of one or several of these three components in the case of music sounds [1,8,14,18,20]. The use of MM for feature extraction in spectrograms has also been done, but rather in speech spectrograms [3,17,22]. A similar approach is used for detecting lines in videos [6]. The first use of MM for analyzing music sounds is [12].

3 Spectrograms

In this section we expose the main formulas to generate spectrograms, following the book [5]. We expose them in a continuous framework such that they remain general and elegant. However, to apply morphological operators, we sample the result on a uniform grid with a specific time-frequency resolution[3].

[3] In the experiments exposed in this work, we chose a 10 ms step for time and a $\frac{44100}{4096} \approx 10.77$ Hz step for frequency. These values are common values for music applications.

For our applications, we focus on the spectrogram of the Short-time Fourier transform (STFT). The STFT is defined as follows: let $f \in \mathcal{L}_\infty(\mathbb{R};\mathbb{C})$ and let $g \in \mathcal{L}_1(\mathbb{R};\mathbb{C})$. The STFT of f with respect to the window g is defined by

$$\begin{array}{rl} \mathrm{STFT}_g[f] : \mathbb{R}\times\mathbb{R} \to \mathbb{C} & \\ (t,\xi) \;\mapsto \int_{\mathbb{R}} f(x)\overline{g(x-t)}e^{-2\pi i t\xi}\,dx & \end{array}, \tag{1}$$

with i being the imaginary unit and $\overline{g(x-t)}$ the complex conjugate of $g(x-t)$.

The STFT is then a functional operator (it takes a function as input and returns a function as output). The input function f is associated to the input signal (an audio signal in our case) whose variable t represents time. The resulting function $\mathrm{STFT}_g[f]$ can be interpreted as the Fourier transform of a neighborhood of the function f around t (the neighborhood being determined by the window function g). The time will be measured in seconds (abbreviated s) and the frequency will be measured in Hertz (abbreviated Hz).

The function $\mathrm{STFT}_g[f]$ has as domain the time-frequency plane $\mathbb{R}\times\mathbb{R}$ and as codomain $\mathbb{C}$. Complex values are difficult to handle and to represent. In practical applications, we usually drop the phase information of the complex value and keep only the amplitude. This is what happens when using a spectrogram, that is defined as

$$\mathrm{SPEC}_g = |\mathrm{STFT}_g|^2 . \tag{2}$$

This leads to an interpretation of the spectrogram as a power value: its value at a time-frequency point (t,ξ) is the power of frequency ξ at time t. We commonly use the logarithmic scale to represent it and thus measure it in decibels, *i.e.*

$$|\mathrm{SPEC}_g|^2 = 10\log_{10}|\mathrm{SPEC}_g|^2\,\mathrm{dB} = 20\log_{10}|\mathrm{SPEC}_g|\,\mathrm{dB} . \tag{3}$$

Since the input signal is bounded by 1 and we normalize the window function, we have that $0 \le |\mathrm{SPEC}_g|^2 \le 1$, which turns out to

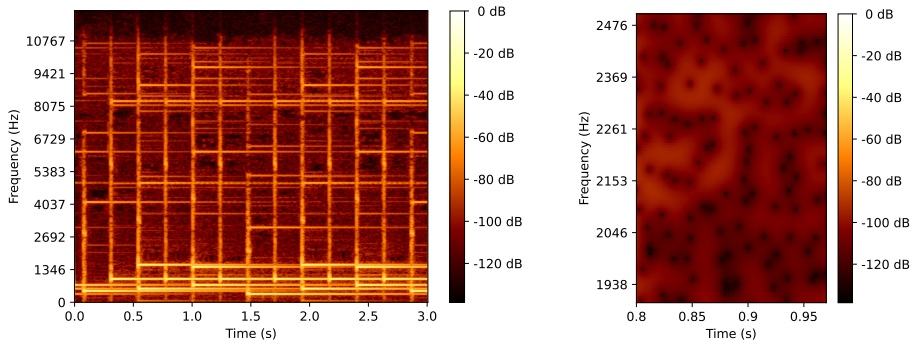
$$-\infty \le 20\log_{10}|\mathrm{SPEC}_g|\,\mathrm{dB} \le 0 \tag{4}$$

in logarithmic scale, obtaining as codomain $\overline{\mathbb{R}}$.

Figure 1 shows the spectrogram of an audio excerpt that will be used as example; we can see the horizontal lines corresponding to sinusoids and the vertical lines corresponding to transients. We can also see the landscape of hills and holes for the noise component.

4 Morphological pipeline

Based on a pipeline of simple MM operators, we propose a method for exploiting the structure of the spectrogram and the geometry of its components to achieve the detection of the three mentioned components. Let us expose the pipeline and illustrate it with the spectrogram from Figure 1.



(a) The sinusoidal and transient component appear as horizontal and vertical lines

(b) The noise component appears as a landscape of hills and holes

Fig. 1: Spectrogram of an audio excerpt.

The pipeline is decomposed into three steps: pre-processing, processing and post-processing. All three steps use MM operators. The pre-processing is performed in order to adapt the image for the processing operations, and the main idea is to *fill the holes* of the spectrogram to have an image that is well adapted for the processing step. In the processing step, we use three different chains, one per component. Finally, for the components associated with lines (the sinusoidal and transient component) we use a post-processing step that allows removing small lines that might appear as residuals. The full morphological pipeline is exposed in Figure 2.

4.1 Pre-processing

The main goal of the pre-processing is to *fill the holes* of the spectrogram. These holes correspond to the zeros of the spectrogram ($-\infty$ in logarithmic scale) and are present when there is noise.

To do that, we apply first a closing and then a reconstruction by erosion. The closing writes, as usual [2,11,13], $\varphi_B = \varepsilon_B \circ \delta_B$ where the dilation δ_B and the erosion ε_B are defined for a function $S \in \overline{\mathbb{R}}^{\mathbb{R}\times\mathbb{R}}$ and a structuring element $B \subseteq \mathbb{R} \times \mathbb{R}$ as $\delta_B[S] = S \oplus B$ and $\varepsilon_B[S] = S \ominus B$ with

$$\begin{aligned} S \oplus B : \mathbb{R}\times\mathbb{R} &\to \overline{\mathbb{R}} \\ p &\mapsto \bigvee_{x\in B} S(p-x) \end{aligned} \qquad \begin{aligned} S \ominus B : \mathbb{R}\times\mathbb{R} &\to \overline{\mathbb{R}} \\ p &\mapsto \bigwedge_{x\in B} S(p+x) \end{aligned} . \tag{5}$$

We use a closing with a structuring element that is a square of width 25 ms and height 75 Hz[4]. These values have been determined experimentally such that they are sufficiently large for filling the holes.

[4] These continuous values are sampled according to the grid, and become 7×3 in our case.

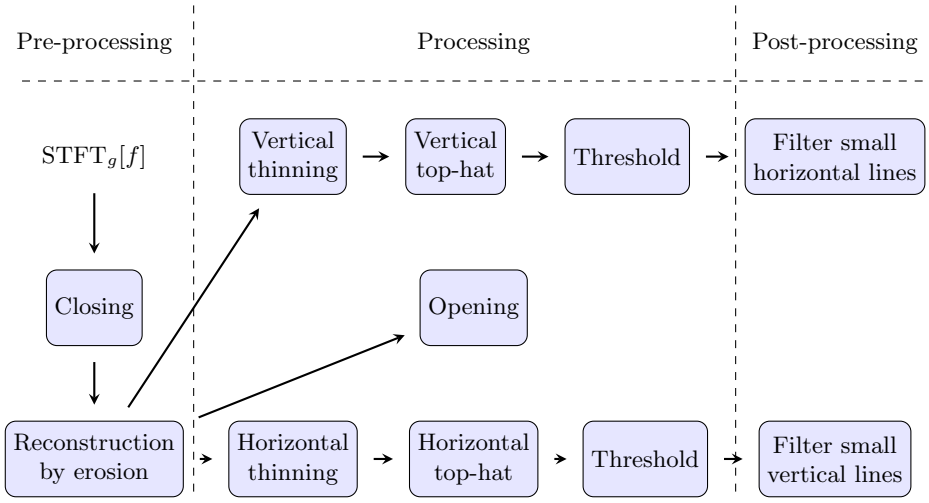


Fig. 2: Pipeline for the morphological processing.

While the closing manages to fill the holes (if B is sufficiently big), it deforms the shapes. To avoid this effect, we apply as next step a reconstruction by erosion. The reconstruction by erosion is defined as follows: we choose an image S to be the marker and we choose an image R to be the reference. Then, the geodesic erosion is defined as

$$\varepsilon_{R,B}[S] = \varepsilon_B[S] \vee R \tag{6}$$

where B is the structuring element of radius 1. The reconstruction by erosion is thus defined as the iteration of this operator until stability, *i.e.*

$$\varepsilon^{\infty}_{R,B}[S] = \varepsilon^{(n)}_{R,B}[S] \tag{7}$$

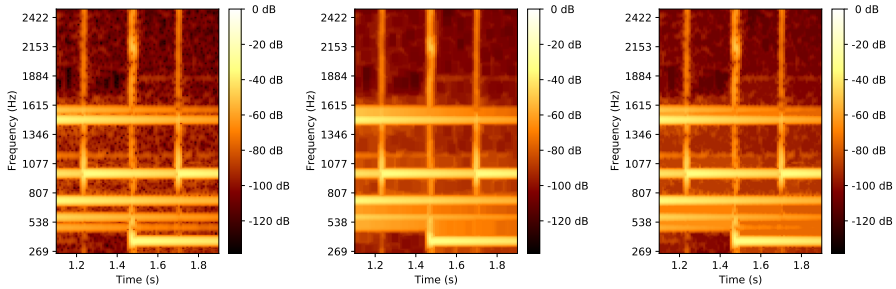
where $\varepsilon^{(n)}_{R,B}[S] = (\underbrace{\varepsilon_{R,B} \circ \cdots \circ \varepsilon_{R,B}}_{n \text{ times}})[S]$ and we have $\varepsilon^{(n)}_{R,B}[S] = \varepsilon^{(n+1)}_{R,B}[S]$.

For our purpose, we use as marker the result of the closing and as reference the input image. We use a structuring element B corresponding to the 8-connectivity, *i.e.* a square of size 3×3.

The results of our pre-processing on our example are shown in Figure 3; we see that the holes disappear in the closing step but with a deformation of the shapes, and the reconstruction by erosion recovers the shapes of the lines.

4.2 Extracting the noise mask

Now that we have an input image without holes, we can apply our processing pipeline to recover the mask of the noise and the lines. Let us start by exposing how to recover the mask of the noise since it is the simpler part as it only involves a single operation.

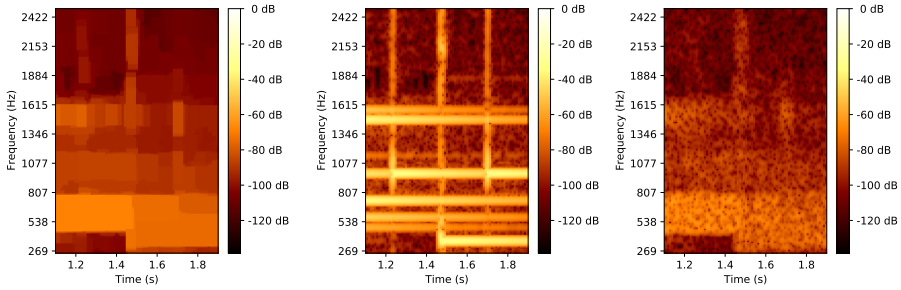


(a) Input (b) Closing (c) Reconstruction

Fig. 3: Pre-processing of our excerpt.

The way we recover the mask of the noise is by applying an opening $\gamma_B = \delta_B \circ \varepsilon_B$ to the pre-processed image. We use as structuring element a square of width 44 ms and height 193 Hz which are the values that ensure a -60 dB drop both in time and frequency in the shape of the window function.

After recovering the mask of the noise, we can filter a white noise with it by pixel-wise multiplication and produce a filtered noise that is similar to the one present in the input spectrogram. This can be remarked in Figure 4.



(a) Opening (b) Input (c) Filtered noise

Fig. 4: Processing of the noise component.

4.3 Recovering the lines

The process of recovering horizontal and vertical lines are dual; we use the same operation but with different structuring elements. For the horizontal lines, we apply a vertical thinning, followed by a vertical top-hat and finally a threshold.

For the vertical lines, we apply a horizontal thinning, followed by a horizontal top-hat and finally also a threshold. Let us describe these three operations independently of the kind of lines we search for.

Thinning The first step is to thin the pre-processed image; this way we obtain lines that are one-pixel thin. However, since our image is greyscale, we shall use a greyscale approach. We apply the one presented in [4] based on the notion of destructible point. From an implementation point of view, this is equivalent to apply a hit-or-miss transform with some particular structuring elements.

The hit-or-miss transform that we are using is one of the ones presented in [10] and is defined as follows: for an input image S and a pair of structuring elements C and D, the hit-or-miss transform $S \circledast (C, D)$ is defined for all $p \in \mathbb{R} \times \mathbb{R}$ as

$$(S \circledast (C, D))(p) = \big((S \ominus C)(p) - (S \oplus \check{D})(p)\big) \vee 0 \tag{8}$$

where $\check{D} = \{-d \in \mathbb{R} \times \mathbb{R} : d \in D\}$.

Now, we can define the elementary thinning of S by C and D as

$$S \circ (C, D) = S - (S \circledast (C, D))\,. \tag{9}$$

We call (C, D) a template.

Applying successive elementary thinnings

$$(((S \circ (C_1, D_1)) \circ (C_2, D_2)) \circ \cdots) \circ (C_n, D_n) \tag{10}$$

is called a thinning and is denoted as

$$S \circ \big((C_1, D_1), (C_2, D_2), \cdots, (C_n, D_n)\big)\,. \tag{11}$$

If we apply the thinning operation until stability, we have an ultimate thinning.

The type of thinning we apply depends on the templates we choose. In our case, we want to perform two different types of thinning: one that thins the lines vertically (and then produces horizontal lines of one pixel length) and another that thins the lines horizontally (and then produces vertical lines of one pixel length).

For the vertical thinning we use the following templates:

$$(C, D)_N,\ (C, D)_S,\ (C, D)_{NE},\ (C, D)_{SW},\ (C, D)_{NW},\ (C, D)_{SE} \tag{12}$$

in this (arbitrary) order, and for the horizontal thinning we use the templates

$$(C, D)_E,\ (C, D)_W,\ (C, D)_{NW},\ (C, D)_{SE},\ (C, D)_{NE},\ (C, D)_{SW} \tag{13}$$

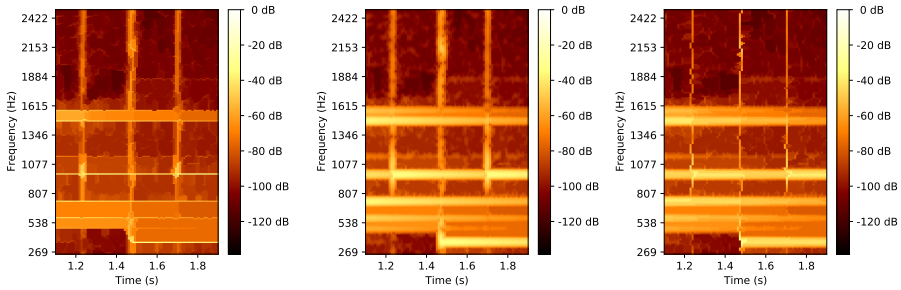
in this (arbitrary) order, where the templates corresponding to each cardinal point are

$$N: \begin{bmatrix} 0 & 0 & 0 \\ - & 1 & - \\ - & 1 & - \end{bmatrix} \qquad E: \begin{bmatrix} - & - & 0 \\ 1 & 1 & 0 \\ - & - & 0 \end{bmatrix} \qquad S: \begin{bmatrix} - & 1 & - \\ - & 1 & - \\ 0 & 0 & 0 \end{bmatrix} \qquad W: \begin{bmatrix} 0 & - & - \\ 0 & 1 & 1 \\ 0 & - & - \end{bmatrix}$$

$$NE: \begin{bmatrix} - & 0 & 0 \\ 1 & 1 & 0 \\ - & 1 & - \end{bmatrix} \qquad SE: \begin{bmatrix} - & 1 & - \\ 1 & 1 & 0 \\ - & 0 & 0 \end{bmatrix} \qquad SW: \begin{bmatrix} - & 1 & - \\ 0 & 1 & 1 \\ 0 & 0 & - \end{bmatrix} \qquad NW: \begin{bmatrix} 0 & 0 & - \\ 0 & 1 & 1 \\ - & 1 & - \end{bmatrix}$$

These patterns should be interpreted in the following manner: ones correspond to the elements of set C, zeroes correspond to the elements of set D, and $-$ means that the point is not considered in the structuring element. The origin is located at the center pixel.

The result of the two types of thinnings are shown in Figure 5; we see that the lines have been reduced to one pixel width but only in one direction.



(a) Vertical thinning (b) Input (c) Horizontal thinning

Fig. 5: Thinnings for reduce the lines to one pixel width.

Top-hat The top-hat operation allows us now to isolate these lines of one pixel width. Similarly as in the case of thinning, we use a type of top hat for each type of thinning (vertical and horizontal). The top hat operation is defined as: $\forall p \in \mathbb{R} \times \mathbb{R}$,

$$S(p) - \gamma_B[S](p) \tag{14}$$

where γ_B is an opening with structuring element $B \subseteq \mathbb{R} \times \mathbb{R}$.

The structuring elements we choose are 3×1 and 1×3 for vertical and horizontal top-hats, respectively.

Threshold After applying the top-hat, we want to recover the values that are above a certain threshold (that we set to 5 dB in our applications). The amplitude values we recover are not the ones of the top-hat but the ones of the result of the pre-processing, to be able to have the correct amplitudes. The operation is described as follows: if we denote the result of the pre-processing as S_0 and the result of the top-hat as $S_{\mathbf{Id}-\gamma}$, the threshold $S_>$ is defined for all $p \in \mathbb{R} \times \mathbb{R}$ as

$$S_>(p) = \begin{cases} S_0(p) & \text{if } S_{\mathbf{Id}-\gamma}(p) > \tau \\ -\infty & \text{if } S_{\mathbf{Id}-\gamma}(p) \leq \tau \end{cases} \tag{15}$$

4.4 Post-processing

With this processing we manage to isolate the horizontal and vertical lines. However, we get a lot of residuals that are too small to represent actual sinusoids and transients. This is why we apply a post-processing operation to remove these small lines.

The operation consists of two steps: first, we apply a thinning that shrinks the lines and eventually make them disappear (if they are below a threshold in length). Then, we apply a reconstruction by dilation with the thinned image as marker and the processed image as reference. This way, the lines that are sufficiently long will reappear in their full length.

Thinning The thinning is done now in the same direction as the lines: we apply an horizontal thinning for the horizontal lines and a vertical thinning for the vertical ones. We actually only use the templates $(C, D)_E$ and $(C, D)_W$ for the horizontal thinning and $(C, D)_N$ and $(C, D)_S$ for the vertical one. The number of iterations is fixed to eliminate lines of a certain width or height; if τ_{s} (resp. τ_{Hz}) is the threshold for horizontal lines and Δt (resp. $\Delta\xi$) is the time (resp. frequency) resolution of the spectrogram, the number of iterations N_{s} (resp. N_{Hz}) is given by $N_{\mathrm{s}} = \left\lceil \frac{\tau_{\mathrm{s}}}{2\Delta t} \right\rceil$ and $N_{\mathrm{Hz}} = \left\lceil \frac{\tau_{\mathrm{Hz}}}{2\Delta\xi} \right\rceil$.

Reconstruction by dilation The reconstruction by dilation is the dual of the reconstruction by erosion. The geodesic dilation is defined as

$$\delta_{R,B}[S] = \delta_B[S] \wedge R \tag{16}$$

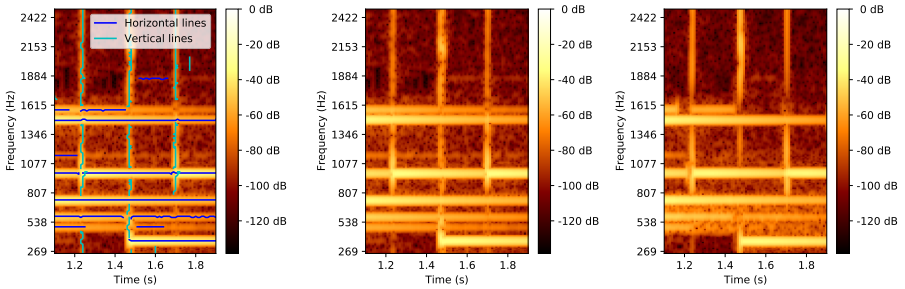
where S is the marker and R the reference. The reconstruction by dilation is the iteration of the geodesic dilation until stability and is denoted by $\delta^{\infty}_{R,B}[S]$. We use a structuring element B corresponding to the 8-connectivity, *i.e.* a square of 3×3.

At the end of the process, we recover a number of lines with which we can re-synthesize the input signal. Figure 6 shows the result on our excerpt, that yields very good results.

5 Experimental results

In this section, we present the outcomes of our experiments, which highlight the effectiveness of our proposed pipeline. We acknowledge that while our chosen example features a glockenspiel (a musical instrument known for its distinct transient and sinusoidal components) the applicability of our pipeline may vary across different instruments.

Our pipeline was applied to a range of musical instruments, resulting in diverse outcomes (Figure 7). In most cases, our pipeline performed considerably good without the need for parameter adjustments. However, there were instances



(a) Recovered lines (b) Input spectrogram (c) Re-synthesis

Fig. 6: Result of the processing in the case of our example.

where fine-tuning was necessary to achieve optimal results. A deeper analysis of these parameters and quantitative evaluation are left for future work.

In scenarios involving certain instruments or complex musical compositions with multiple instruments, the separation of the components (sinusoids, transients, and noise) proved challenging. This often resulted in sparse recovered lines, with the noise component dominating the information.

We performed the operations using the Python libraries NumPy [7], SciPy [19] and scikit-image [21]. However, greyscale thinning was not implemented in these libraries. Consequently, we implemented it through iterative processing, which became the bottleneck of our pipeline.

Furthermore, initially, we opted for a time resolution of 1 ms, resulting in very large images that noticeably slowed down the process. In an effort to improve efficiency, we experimented with increasing the time resolution to 10 ms. Remarkably, this change did not negatively impact the results while significantly enhancing the processing speed.

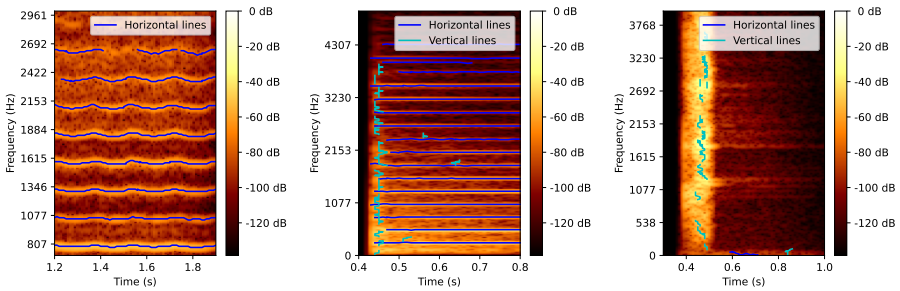
Audio examples and all the code used for the processing the spectrograms and generating the images are available at `https://github.com/Manza12/DGMM-2024`.

6 Conclusions and future work

In this work, we have proposed a morphological pipeline for extracting components from spectrograms of musical audio signals. We have shown how to detect the lines that correspond to sinusoids and transients through the use of a morphological pipeline, as well as how to extract a mask for the noise component.

While in general the method works well for simple excerpts, we noticed that when we apply it to very dense spectrograms it performs badly. This is probably caused by a difficulty in deciding if something is a line or if it belongs to the noise component in such situations.

While working on this topic, we found that the border between noise and signal (understanding signal as sinusoids or transients) is fuzzy. In future work



(a) Violin (b) Piano (c) Güiro

Fig. 7: Lines detection for other instruments.

we will go more in depth about the nature of spectrograms, proposing another model different from the STN model and more adapted to the detection ridges and holes, and making a more clear distinction between noise and signal.

References

1. Amatriain, X., Bonada, J., Loscos, A., Serra, X.: Spectral Processing. In: DAFX, chap. 10, pp. 373–438. John Wiley & Sons, Ltd (2002)
2. Bloch, I., Heijmans, H., Ronse, C.: Mathematical Morphology. In: Handbook of Spatial Logics, pp. 857–944. Springer, Dordrecht, The Netherlands (2007)
3. Cadore, J., Gallardo-Antolín, A., Peláez-Moreno, C.: Morphological Processing of Spectrograms for Speech Enhancement. In: 5th International Conference on Nonlinear Speech Processing. pp. 224–231. Berlin, Heidelberg (2011)
4. Couprie, M., Bezerra, F.N., Bertrand, G.: Topological operators for grayscale image processing. Journal of Electronic Imaging **10**(4), 1003–1015 (2001)
5. Gröchenig, K.: Foundations of Time-Frequency Analysis. Birkhäuser, Boston (2001)
6. Guimarães, S.J.F., Couprie, M., de Albuquerque Araújo, A., Jerônimo Leite, N.: Video segmentation based on 2D image analysis. Pattern Recognition Letters **24**, 947–957 (2003)
7. Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del Río, J.F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E.: Array programming with NumPy. Nature **585**(7825), 357–362 (2020)
8. Keiler, F., Marchand, S.: Survey on Extraction of Sinusoids in Stationary Sounds. In: Digital Audio Effects (DAFx) Conference. pp. 51–58. Germany (2002)
9. Klapuri, A., Davy, M.: Signal Processing Methods for Music Transcription. Springer Science & Business Media (2007)
10. Naegel, B., Passat, N., Ronse, C.: Grey-Level Hit-or-Miss Transforms—Part I: Unified Theory. Pattern Recognition **40**(2), 635–647 (2007)

11. Najman, L., Talbot, H.: Mathematical Morphology: From Theory to Applications. Wiley-ISTE, London (2010)
12. Romero-García, G., Agón, C., Bloch, I.: Estimation de paramètres de resynthèse de sons d'instruments de musique avec des outils de morphologie mathématique. In: 19th Sound and Music Computing Conference. pp. 653–662. Zenodo, Saint-Etienne, France (2022)
13. Ronse, C., Heijmans, H.J.A.M.: The Algebraic Basis of Mathematical Morphology: II. Openings and Closings. CVGIP: Image Understanding **54**(1), 74–97 (1991)
14. Salamon, J., Gomez, E.: Melody Extraction From Polyphonic Music Signals Using Pitch Contour Characteristics. IEEE Transactions on Audio, Speech, and Language Processing **20**(6), 1759–1770 (2012)
15. Serra, X.: Musical Sound Modeling with Sinusoids plus Noise. In: Musical Signal Processing, pp. 91–122. Routledge, New York, USA (1997)
16. Serra, X., Smith, J.: Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic Plus Stochastic Decomposition. Computer Music Journal **14**(4), 12–24 (1990)
17. Steinberg, R., O'Shaughnessy, D.: Segmentation of a speech spectrogram using mathematical morphology. In: IEEE International Conference on Acoustics, Speech and Signal Processing. pp. 1637–1640 (2008)
18. Verma, T.S., Meng, T.H.Y.: Extending Spectral Modeling Synthesis with Transient Modeling Synthesis. Computer Music Journal **24**(2), 47–59 (2000)
19. Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., SciPy 1.0 Contributors: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods **17**, 261–272 (2020). https://doi.org/10.1038/s41592-019-0686-2
20. Virtanen, T., Klapuri, A.: Separation of harmonic sound sources using sinusoidal modeling. In: IEEE International Conference on Acoustics, Speech, and Signal Processing. pp. II765–II768 (2000)
21. Van der Walt, S., Schönberger, J.L., Nunez-Iglesias, J., Boulogne, F., Warner, J.D., Yager, N., Gouillart, E., Yu, T.: scikit-image: image processing in python. PeerJ **2**, e453 (2014)
22. Xu, S., Zhao, X., Duan, C.H., Cao, X.L., Li, H.Y., Liang, S.L., Wang, S.W.: A Mathematical Morphological Processing of Spectrograms for the Tone of Chinese Vowels Recognition. Applied Mechanics and Materials **571–572**, 665–671 (2014)